CICS Transaction Server for z/OS



API Reference

Version 5 Release 4

CICS Transaction Server for z/OS



API Reference

Version 5 Release 4

Note
Before using this information and the product it supports, read the information in "Notices" on page 891.

This edition applies to the IBM CICS Transaction Server for z/OS Version 5 Release 4 (product number 5655-Y04) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 1989, 2017.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this PDF vii

Chapter 1. CICS API command format . . 1

CICS command syntax notation 2
CICS command argument values 3
CICS command restrictions 10
LENGTH options in CICS commands 10
NOHANDLE option 10
RESP and RESP2 options 10
Translated code for CICS commands 11
COBOL translation output 12
C translation output 12
PL/I translation output 12
Assembler translation output 13

Chapter 2. CICS command summary 17

ABEND 25
ACQUIRE 27
ADD SUBEVENT 29
ADDRESS 31
ADDRESS SET 33
ALLOCATE (APPC) 33
ALLOCATE (LUTYPE6.1). 37
ALLOCATE (MRO). 38
ASKTIME 40
ASSIGN 41
Codes returned by ASSIGN 57
BIF DEEDIT 59
BIF DIGEST 60
BUILD ATTACH (LUTYPE6.1) 61
BUILD ATTACH (MRO) 64
CANCEL 66
CANCEL (BTS) 68
CHANGE PHRASE. 70
CHANGE PASSWORD 72
CHANGE TASK 74
CHECK ACQPROCESS 74
CHECK ACTIVITY 76
CHECK TIMER 79
CONNECT PROCESS 80
CONVERSE (default) 83
CONVERSE (APPC) 83
CONVERSE (LUTYPE2/LUTYPE3) 83
CONVERSE (LUTYPE4) 84
CONVERSE (LUTYPE6.1) 84
CONVERSE (SCS) 85
CONVERSE (3270 logical) 86
CONVERSE (3600-3601) 86
CONVERSE (3600-3614) 87
CONVERSE (3650 interpreter) 88
CONVERSE (3650-3270) 88
CONVERSE (3650-3653) 89
CONVERSE (3650-3680) 89
CONVERSE (3767) 90
CONVERSE (3770) 90
CONVERSE (3790 full-function or inquiry) 91
CONVERSE (3790 3270-display) 91
CONVERSE: z/OS Communications Server options 91
CONVERSE (non-z/OS Communications Server default). 96
CONVERSE (MRO). 97
CONVERSE (2260) 97
CONVERSE: non-z/OS Communications Server options 97
CONVERTTIME 102
DEFINE ACTIVITY 104
DEFINE COMPOSITE EVENT. 106
DEFINE COUNTER and DEFINE DCOUNTER . . 108
DEFINE INPUT EVENT 111
DEFINE PROCESS 112
DEFINE TIMER 115
DELAY 117
DELETE 120
DELETE ACTIVITY 128
DELETE CHANNEL 129
DELETE CONTAINER (BTS) 130
DELETE CONTAINER (CHANNEL) 132
DELETE COUNTER and DELETE DCOUNTER 133
DELETE EVENT 135
DELETE TIMER 136
DELETEQ TD 136
DELETEQ TS 138
DEQ 139
DOCUMENT CREATE 141
DOCUMENT DELETE 145
DOCUMENT INSERT 146
DOCUMENT RETRIEVE 149
DOCUMENT SET 151
DUMP TRANSACTION 153
ENDBR 158
ENDBROWSE ACTIVITY 160
ENDBROWSE CONTAINER (BTS) 161
ENDBROWSE CONTAINER (CHANNEL). . . . 161
ENDBROWSE EVENT 162
ENDBROWSE PROCESS 163
ENQ 163
ENTER TRACENUM. 166
EXTRACT ATTACH (LUTYPE6.1) 168
EXTRACT ATTACH (MRO) 171
EXTRACT ATTRIBUTES (APPC) 174
EXTRACT ATTRIBUTES (MRO) 175
EXTRACT CERTIFICATE 177
EXTRACT LOGONMSG. 179
EXTRACT PROCESS 180
EXTRACT TCPIP 182
EXTRACT TCT. 185
EXTRACT WEB 186
| FETCH ANY 191
| FETCH CHILD 192
FORCE TIMER 194
FORMATTIME 195

© Copyright IBM Corp. 1989, 2017

FREE 200
FREE (APPC) 201
FREE (LUTYPE6.1) 202
FREE (MRO) 203
| FREE CHILD 204
FREEMAIN 205
FREEMAIN64 208
GDS ALLOCATE 210
GDS ASSIGN 212
GDS CONNECT PROCESS 213
GDS EXTRACT ATTRIBUTES 215
GDS EXTRACT PROCESS 216
GDS FREE 218
GDS ISSUE ABEND 219
GDS ISSUE CONFIRMATION 221
GDS ISSUE ERROR 222
GDS ISSUE PREPARE 223
GDS ISSUE SIGNAL 225
GDS RECEIVE 226
GDS SEND 228
GDS WAIT 230
GET CONTAINER (BTS) 232
GET CONTAINER (CHANNEL) 234
GET COUNTER and GET DCOUNTER 239
GETMAIN 244
GETMAIN64 248
GETNEXT ACTIVITY 252
GETNEXT CONTAINER (BTS) 253
GETNEXT CONTAINER (CHANNEL) 254
GETNEXT EVENT 255
GETNEXT PROCESS 257
GET64 CONTAINER 258
HANDLE ABEND. 262
HANDLE AID 264
HANDLE CONDITION 267
IGNORE CONDITION 268
INQUIRE ACTIVITYID 269
INQUIRE CONTAINER 272
INQUIRE EVENT 274
INQUIRE PROCESS 275
INQUIRE TIMER 276
INVOKE APPLICATION 278
INVOKE SERVICE 281
INVOKE WEBSERVICE 286
ISSUE ABEND 287
ISSUE ABORT 288
ISSUE ADD 290
ISSUE CONFIRMATION 292
ISSUE COPY (3270 logical) 293
ISSUE DISCONNECT (default) 294
ISSUE DISCONNECT (LUTYPE6.1) 296
ISSUE END 297
ISSUE ENDFILE 298
ISSUE ENDOUTPUT 299
ISSUE EODS 300
ISSUE ERASE 301
ISSUE ERASEAUP 303
ISSUE ERROR 304
ISSUE LOAD 305
ISSUE NOTE 306
ISSUE PASS 308
ISSUE PREPARE 309
ISSUE PRINT 311
ISSUE QUERY 312
ISSUE RECEIVE 313
ISSUE REPLACE 315
ISSUE RESET 317
ISSUE SEND 317
ISSUE SIGNAL (APPC) 319
ISSUE SIGNAL (LUTYPE6.1) 321
ISSUE WAIT. 322
JOURNAL 323
LINK 323
LINK ACQPROCESS 332
LINK ACTIVITY 335
LOAD 338
MONITOR 341
MOVE CONTAINER (BTS) 343
MOVE CONTAINER (CHANNEL) 345
POINT 348
POP HANDLE 349
POST 349
PURGE MESSAGE 353
PUSH HANDLE 354
PUT CONTAINER (BTS) 354
PUT CONTAINER (CHANNEL) 357
PUT64 CONTAINER 361
QUERY CHANNEL 365
QUERY COUNTER and QUERY DCOUNTER . . 365
QUERY SECURITY 369
READ 372
READNEXT 384
READPREV 395
READQ TD 405
READQ TS 409
RECEIVE (z/OS Communications Server default) 412
RECEIVE (APPC) 413
RECEIVE (LUTYPE2/LUTYPE3) 413
RECEIVE (LUTYPE4) 414
RECEIVE (LUTYPE6.1) 414
RECEIVE (3270 logical) 415
RECEIVE (3600 pipeline) 415
RECEIVE (3600-3601) 416
RECEIVE (3600-3614) 416
RECEIVE (3650) 417
RECEIVE (3767) 417
RECEIVE (3770) 418
RECEIVE (3790 full-function or inquiry) 418
RECEIVE: z/OS Communications Server options 419
RECEIVE (non-z/OS Communications Server default) 422
RECEIVE (MRO) 422
RECEIVE (2260) 423
RECEIVE (2980) 424
RECEIVE (3790 3270-display) 426
RECEIVE: non-z/OSCommunications Server options 426
RECEIVE MAP 430
RECEIVE MAP MAPPINGDEV 433
RECEIVE PARTN 435
RELEASE. 438
REMOVE SUBEVENT 439

REQUEST ENCRYPTPTKT 440
REQUEST PASSTICKET 443
RESET ACQPROCESS 444
RESET ACTIVITY 446
RESETBR 447
RESUME 451
RETRIEVE 453
RETRIEVE REATTACH EVENT 456
RETRIEVE SUBEVENT 458
RETURN 459
REWIND COUNTER and REWIND DCOUNTER 463
REWRITE 465
ROUTE 471
RUN 475
| RUN TRANSID 479
SEND (z/OS Communications Server default) . . 481
SEND (APPC) 481
SEND (LUTYPE2/LUTYPE3) 482
SEND (LUTYPE4) 482
SEND (LUTYPE6.1) 483
SEND (SCS) 483
SEND (3270 logical) 484
SEND (3600 pipeline) 484
SEND (3600-3601) 485
SEND (3600-3614) 486
SEND (3650 interpreter) 486
SEND (3650-3270) 487
SEND (3650-3653) 487
SEND (3650-3680) 488
SEND (3767) 488
SEND (3770) 488
SEND (3790 full-function or inquiry) 489
SEND (3790 SCS) 489
SEND (3790 3270-display) 490
SEND (3790 3270-printer) 490
SEND: z/OS Communications Server options . . 490
SEND (non-z/OS Communications Server default) 494
SEND (MRO) 495
SEND (2260) 495
SEND (2980) 495
SEND: non-z/OS Communications Server options 496
SEND CONTROL 499
SEND MAP 504
SEND MAP MAPPINGDEV 511
SEND PAGE 514
SEND PARTNSET 517
SEND TEXT 518
SEND TEXT MAPPED 525
SEND TEXT NOEDIT 527
SIGNAL EVENT 530
SIGNOFF 532
SIGNON 533
SIGNON TOKEN 537
SOAPFAULT ADD 541
SOAPFAULT CREATE 543
SOAPFAULT DELETE 546
SPOOLCLOSE 547
SPOOLOPEN INPUT 549
SPOOLOPEN OUTPUT 552
SPOOLREAD 556
SPOOLWRITE 559
START 561
START ATTACH 572
START BREXIT 573
START CHANNEL 576
STARTBR 580
STARTBROWSE ACTIVITY 587
STARTBROWSE CONTAINER (BTS) 588
STARTBROWSE CONTAINER (CHANNEL) . . . 590
STARTBROWSE EVENT 591
STARTBROWSE PROCESS 592
SUSPEND 593
SUSPEND (BTS) 593
SYNCPOINT 595
SYNCPOINT ROLLBACK 596
TEST EVENT 597
| TRANSFORM DATATOJSON 597
TRANSFORM DATATOXML 599
| TRANSFORM JSONTODATA 602
TRANSFORM XMLTODATA 603
UNLOCK 606
UPDATE COUNTER and UPDATE DCOUNTER 610
VERIFY PASSWORD 613
VERIFY PHRASE 616
VERIFY TOKEN 619
WAIT CONVID (APPC) 623
WAIT EVENT 624
WAIT EXTERNAL 626
WAIT JOURNALNAME 628
WAIT JOURNALNUM 630
WAIT SIGNAL 631
WAIT TERMINAL 632
WAITCICS 633
WEB CLOSE 635
WEB CONVERSE 637
WEB ENDBROWSE FORMFIELD 650
WEB ENDBROWSE HTTPHEADER 651
WEB ENDBROWSE QUERYPARM 651
WEB EXTRACT 652
WEB OPEN 657
WEB PARSE URL 662
WEB READ FORMFIELD 664
WEB READ HTTPHEADER 667
WEB READ QUERYPARM 668
WEB READNEXT FORMFIELD 670
WEB READNEXT HTTPHEADER 671
WEB READNEXT QUERYPARM 673
WEB RECEIVE (Server) 674
WEB RECEIVE (Client) 681
WEB RETRIEVE 688
WEB SEND (Server) 689
WEB SEND (Client) 698
WEB STARTBROWSE FORMFIELD 708
WEB STARTBROWSE HTTPHEADER 710
WEB STARTBROWSE QUERYPARM 711
WEB WRITE HTTPHEADER 712
WRITE 715
WRITE JOURNALNAME 722
WRITE JOURNALNUM 725
WRITE OPERATOR 726
WRITEQ TD 729
WRITEQ TS 731

WSACONTEXT BUILD 735
WSACONTEXT DELETE 739
WSACONTEXT GET 740
WSAEPR CREATE 744
XCTL 747

Chapter 3. Threadsafe commands . . 751

Chapter 4. EXEC interface block fields 755
EXEC interface block (EIB) response and function codes 774
Response codes of EXEC CICS commands. . . 774
Function codes of EXEC CICS commands . . . 775

Chapter 5. CICS-value data areas used by all commands 787
CVDAs and numeric values in alphabetic sequence 787
CVDA values for the DEVICE option 811
CVDAs and numeric values in numeric sequence 814

Chapter 6. National language codes 839

Chapter 7. Terminal control 841
Commands and options for terminals and logical units 841
Teletypewriter programming 843
Display device operations 844
Print displayed information (ISSUE PRINT) . . 844
Copy displayed information (ISSUE COPY) . . 845
Erase all unprotected fields (ISSUE ERASEAUP) 845
Handle input without data (RECEIVE) 846

Chapter 8. Common Programming Interface Communications (CPI Communications) 847
SAA Resource Recovery 847

Chapter 9. BMS-related constants . . 849
Magnetic slot reader (MSR) control value constants, DFHMSRCA 852
MSR control byte values. 852
Attention identifier constants, DFHAID 853

Chapter 10. BMS macros 855
Map set, map, and field definition 855
Partition set definition 856
Field groups. 856
DFHMDF 857
DFHMDI 868
DFHMSD. 877
DFHPDI 888
DFHPSD 889

Notices 891

Index 897

About this PDF

This PDF is a reference of the commands of the CICS® application programming interface. This documentation is intended for application programmers who are writing applications to interact with CICS. Before CICS TS V5.4, this PDF was called the *Application Programming Reference*.

For information about how to write applications for CICS, using this API, see *Developing CICS Applications*.

For details of the terms and notation used in this book, see Conventions and terminology used in the CICS documentation in IBM Knowledge Center.

Date of this PDF

This PDF was created on June 6th 2017.

© Copyright IBM Corp. 1989, 2017

Chapter 1. CICS API command format

The general format of a CICS command is EXECUTE CICS (or EXEC CICS) followed by the name of the required **command**, and possibly by one or more **options**.

The command format is as follows:

```
EXEC CICS command option(arg)....
```

where:

command
: describes the operation required (for example, READ).

option
: describes any of the many optional facilities available with each function. Some options are followed by an argument in parentheses. You can write options (including those that require arguments) in any order.

arg
: **(short for argument)** is a value such as "data-value" or "name". A "data-value" can be a constant, this means that an argument that sends data to CICS is generally a "data-value". An argument that receives data from CICS must be a "data-area".

  Some arguments described as "data-area" can both send and receive data. In these cases, you must ensure that the "data-area" is not in protected storage.

An example of a CICS command is as follows:

```
EXEC CICS READ
          FILE('FILEA')
          INTO(FILEA)
          RIDFLD(KEYNUM)
          UPDATE
```

You must add the appropriate end-of-command delimiter; see "CICS command syntax notation" on page 2.

Note: If you want to add comments against CICS commands, you can do this, in assembler only, by using a period or a comma as a delimiter after the last argument. For example:

```
EXEC CICS ADDRESS EIB(MYUEIB),        @F1A
```

If a period or a comma is used with an EXEC CICS command, the following line must begin between column 2 and column 16, with the continuation character in column 72. The following line cannot start after column 17. If there is no comma or period added, the following line must begin at or after column 2 and end by column 71, with the continuation character in column 72.

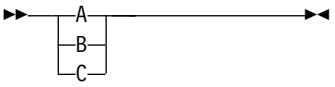
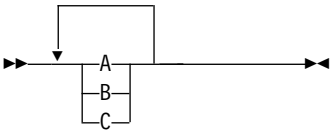
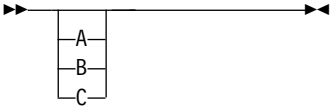
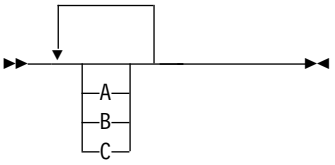
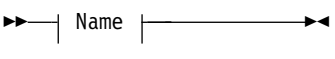
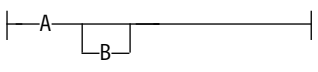
© Copyright IBM Corp. 1989, 2017

CICS command syntax notation

In CICS documentation, CICS commands are presented in a standard way. You interpret the syntax by following the arrows from left to right.

The “EXEC CICS” that always precedes each command's keyword is not included; nor is the “END-EXEC” statement used in COBOL or the semicolon (;) used in PL/I and C that you must code at the end of each CICS command. In the C language, a null character can be used as an end-of-string marker, but CICS does not recognize this; you must therefore never have a comma or period followed by a space (X'40') in the middle of a coding line.

The conventions are:

Symbol	Action
	A set of alternatives—one of which you ***must*** code.
	A set of alternatives—one of which you ***must*** code. You ***may*** code more than one of them, in any sequence.
	A set of alternatives—one of which you ***may*** code.
	A set of alternatives — any number (including none) of which you may code once, in any sequence.
	Alternatives where **A** is the default.
 **Name:** 	Use with the named section in place of its name.
Punctuation and uppercase characters	Code exactly as shown.

Symbol	Action
Lowercase characters	Code your own text, as appropriate (for example, `name`).

CICS command argument values

The data associated with a command option is called its *argument*. Each type of argument can contain different data types; some arguments return information from CICS to the program and others are set by the program.

Options in a CICS command can take the following argument values:

- *data-value*
- *data-area*
- *cvda* (CICS-value data area)
- *ptr-value*
- *ptr-ref*
- *name*
- *filename*
- *systemname*
- *label*
- *hhmmss*

For AMODE(64) programs only, options in a CICS command can also take the following argument values:

- *data-area64*
- *ptr-value64*
- *ptr-ref64*

Data areas and data values

Data areas and data values are the basic argument types. The difference between them is the direction in which information flows when a task executes a command. A *data-value* is always, can only be, a sender; it conveys data to CICS that CICS uses to process the command. A *data-area* is a receiver; CICS uses it to return information to the caller. A *data-area* can also be a sender, for example when the data to convey to CICS is variable length (as in FROM), or where a field is used both for input and output.

COBOL argument values

The argument values can be replaced as follows:

- *data-value* can be replaced by any COBOL data name of the correct data type for the argument, or by a constant that can be converted to the correct type for the argument. The following table shows how to define the correct data type:

Data type	COBOL definition
Halfword binary	PIC S9(4) COMP
Fullword binary	PIC S9(8) COMP
Doubleword *unsigned* binary	PIC 9(18) COMP
Character string	PIC X(*n*) where *n* is the number of bytes

Data type	COBOL definition
UTF-8 character string	PIC X(*n*) where *n* is the number of bytes

- *data-area* can be replaced by any COBOL data name of the correct data type for the argument. The following table shows how to define the correct data type:

Data type	COBOL definition
Halfword binary	PIC S9(4) COMP
Fullword binary	PIC S9(8) COMP
Doubleword *unsigned* binary	PIC 9(18) COMP
Character string	PIC X(*n*) where *n* is the number of bytes
UTF-8 character string	PIC X(*n*) where *n* is the number of bytes

  Where the data type is unspecified, *data-area* can refer to an elementary or group item.
- *cvda* is described in CICS-value data areas (CVDAs).
- *ptr-value* can be replaced by a pointer variable or ADDRESS special register.
- *ptr-ref* can be replaced by a pointer variable or ADDRESS special register.
- *name* can be replaced by either of the following values:
  - A character string specified as an alphanumeric literal. If this string is shorter than the required length, it is padded with blanks.
  - A COBOL data area with a length equal to the required length for the name. The value in *data-area* is the name to be used by the argument. If *data-area* is shorter than the required length, the excess characters are undefined, which might cause unpredictable results.

  filename, as used in FILE(*filename*), specifies the name of the file. The name must contain 1–8 characters from the range A–Z, 0–9, $, @, and #.

  systemname, as used in SYSID(*systemname*), specifies the name of the system to which the request is directed. The name must contain 1–4 characters from the range A–Z, 0–9, $, @, and #.
- *label* can be replaced by any COBOL paragraph name or a section name.
- *hhmmss* can be replaced by a decimal constant or by a data name of the form PIC S9(7) COMP-3. The value must be of the form 0HHMMSS+ where:

  HH Represents hours from 00 through 99.

  MM Represents minutes from 00 through 59.

  SS Represents seconds from 00 through 59.

In COBOL, you do not need to code the LENGTH option unless you want the program to read or write data of a length that is different from that of the referenced variable.

C argument values

The argument values can be replaced as follows:

- *data-value* can be replaced by any C expression that can be converted to the correct data type for the argument. The following table shows how to define the correct data type:

Data type	C definition
Halfword binary	short int
Fullword binary	long int
Doubleword binary	char[8]
Character string	char[*n*] where *n* is the number of bytes
UTF-8 character string	char[*n*] where *n* is the number of bytes

data-value includes *data-area* as a subset.

- *data-area* can be replaced by any C data reference that has the correct data type for the argument. The following table shows how to define the correct data type:

Data type	C definition
Halfword binary	short int
Fullword binary	long int
Doubleword binary	char[8]
Character string	char[*n*] where *n* is the number of bytes
UTF-8 character string	char[*n*] where *n* is the number of bytes

  If the data type is unspecified, *data-area* can refer to a scalar data type, array, or structure. The reference must be to contiguous storage.
- *cvda* is described in CICS-value data areas (CVDAs).
- *ptr-value* (which includes *ptr-ref* as a subset) can be replaced by any C expression that can be converted to an address.
- *ptr-ref* can be replaced by any C pointer type reference.
- *name* can be replaced by either of the following values:
  - A character string in double quotation marks (that is, a literal constant).
  - A C expression or reference whose value can be converted to a character array with a length equal to the maximum length allowed for the name. The value of the character array is the name to be used by the argument.

  filename, as used in FILE(*filename*), specifies the name of the file. The name must have 1–8 characters from the range A–Z, 0–9, $, @, and #.

  systemname, as used in SYSID(*systemname*), specifies the name of the system to which the request is directed. The name must have 1–4 characters from the range A–Z, 0–9, $, @, and #.
- *label* is not supported in the C language.
- *hhmmss* can be replaced by an integer constant; otherwise the application is responsible for ensuring that the value passed to CICS is in packed decimal format. The language does not provide a packed decimal type.

  HH Represents hours from 00 through 99.

  MM Represents minutes from 00 through 59.

  SS Represents seconds from 00 through 59.

Many commands involve the transfer of data between the application program and CICS. In most cases, if SET is used, the LENGTH option must be specified; the syntax of each command and its associated options show whether this rule applies.

PL/I argument values

The argument values can be replaced as follows:

- *data-value* can be replaced by any PL/I expression that can be converted to the correct data type for the argument. The following table shows how to define the correct data type:

Data type	PL/I definition
Halfword binary	FIXED BIN(15)
Fullword binary	FIXED BIN(31)
Doubleword binary	CHAR (8)
Character string	CHAR(*n*) where *n* is the number of bytes
UTF-8 character string	CHAR(*n*) where *n* is the number of bytes

  data-value includes *data-area* as a subset.

- *data-area* can be replaced by any PL/I data reference that has the correct data type for the argument. The following table shows how to define the correct data type:

Data type	PL/I definition
Halfword binary	FIXED BIN(15)
Fullword binary	FIXED BIN(31)
Doubleword binary	CHAR (8)
Character string	CHAR(*n*) where *n* is the number of bytes
UTF-8 character string	CHAR(*n*) where *n* is the number of bytes

  If the data type is unspecified, *data-area* can refer to an element, array, or structure; for example, FROM(P–>STRUCTURE) LENGTH(LNG). The reference must be to connected storage.

  The data area must also have the correct PL/I alignment attribute: ALIGNED for binary items, and UNALIGNED for strings.

  If you use a varying data string without an explicit length, the data passed begins with a two-byte length field, and its length is the maximum length declared for the string. If you explicitly specify a length in the command, the data passed has this length; that is, the two-byte length field followed by data up to the length you specified.

- *cvda* is described in CICS-value data areas (CVDAs).
- *ptr-value* (which includes *ptr-ref*as a subset) can be replaced by any PL/I expression that can be converted to POINTER.
- *ptr-ref* can be replaced by any PL/I reference of type POINTER ALIGNED.
- *name* can be replaced by either of the following values:
  - A character string in single quotation marks (that is, a literal constant).
  - A PL/I expression or reference whose value can be converted to a character string with a length equal to the maximum length allowed for the name. The value of the character string is the name to be used by the argument.

  filename, as used in FILE(*filename*), specifies the name of the file. The name must have 1-8 characters from the range A–Z, 0–9, $, @, and #.

  systemname, as used in SYSID(*systemname*), specifies the name of the system to which the request is directed. The name must have 1-4 characters from the range A–Z, 0–9, $, @, and #.

- *label* can be replaced by any PL/I expression whose value is a label.
- *hhmmss* can be replaced by a decimal constant or an expression that can be converted to a FIXED DECIMAL(7,0). The value must be of the form 0HHMMSS+ where:

  HH Represents hours from 00 through 99.

  MM Represents minutes from 00 through 59.

  SS Represents seconds from 00 through 59.

If the UNALIGNED attribute is added to the ENTRY declarations generated by the CICS translator by a DEFAULT DESCRIPTORS statement, data-area or pointer-reference arguments to CICS commands must also be UNALIGNED. Similarly for the ALIGNED attribute, data-area or pointer-reference arguments must be ALIGNED.

Many commands involve the transfer of data between the application program and CICS. In most cases, the length of the data to be transferred must be provided by the application program. However, if a data area is specified as the source or target, it is not necessary to provide the length explicitly, because the command-language translator generates a default length value of either STG(data-area) or CSTG(data-area), as appropriate.

Assembler-language argument values for AMODE(24) and AMODE(31) programs

In general, an argument can be either the address of the data or the data itself (in assembler language terms, either a relocatable expression or an absolute expression).

A relocatable expression must not contain unmatched brackets (outside quotation marks) or unmatched quotation marks (apart from length-attribute references). If this rule is obeyed, any expression can be used, including literal constants, such as =AL2(100), forms such as 20(0,R11), and forms that use the macro-replacement facilities.

An absolute expression must be a single term that is either a length-attribute reference, or a self-defining constant.

Use care with equated symbols, which should be used only when referring to registers (pointer references). For example, if an equated symbol is used for a length, it is treated as the address of the length and an unpredictable error occurs.

For AMODE(24) and AMODE(31) assembler language programs, the argument values can be replaced as follows:

- *data-value* can be replaced by a relocatable expression that is an assembler-language reference to data of the correct type for the argument, or by a constant of the correct type for the argument.
- *data-area* can be replaced by a relocatable expression that is an assembler-language reference to data of the correct type for the argument.
- *cvda* is described in CICS-value data areas (CVDAs).
- *ptr-value* can be replaced by an absolute expression that is an assembler-language reference to a register.
- *ptr-ref* can be replaced by an absolute expression that is an assembler-language reference to a register.

- *name* can be replaced either by a character string in single quotation marks, or by an assembler-language language relocatable expression reference to a character string. The length is equal to the maximum length allowed for the name. The value of the character string is the name to be used by the argument.

  filename, as used in FILE(*filename)*, specifies the name of the file. The name must have 1–8 characters from the range A–Z, 0–9, $, @, and #.

  systemname, as used in SYSID(*systemname*), specifies the name of the system to which the request is directed. The name must have 1–4 characters from the range A–Z, 0–9, $, @, and #.
- *label* refers to a destination address to which control is transferred. It can be replaced by the label of the destination instruction or by the label of an address constant for the destination. This constant must not specify a length.

  You can also use the expression =A(*dest*) where *dest* is a relocatable expression denoting the destination.

  For example, the following commands are equivalent:

```
HANDLE CONDITION ERROR(DEST)
HANDLE CONDITION ERROR(ADCON)
HANDLE CONDITION ERROR(=A(DEST))
⋮
DEST  BR 14
ADCON DC A(DEST)
```

- *hhmmss* can be replaced by a self-defining decimal constant, or an assembler-language reference to a field defined as PL4. The value must be of the form 0HHMMSS+ where:

  HH Represents hours from 00 through 99

  MM Represents minutes from 00 through 59

  SS Represents seconds from 00 through 59

Many commands involve the transfer of data between the application program and CICS. In most cases, the application program must provide the length of the data to be transferred. However, if a data area is specified as the source or target, it is not necessary to provide the length explicitly, because the command-language translator generates a default length. For example:

```
xxx DC CL8
 .
 .
EXEC CICS ... LENGTH(L'xxx)
```

Assembler-language argument values for AMODE(64) programs

In general, an argument can be either the address of the data or the data itself (in assembler language terms, either a relocatable expression or an absolute expression).

A relocatable expression must not contain unmatched brackets (outside quotation marks) or unmatched quotation marks (apart from length-attribute references). If

this rule is obeyed, any expression can be used, including literal constants, such as =AL2(100), forms such as 20(0,R11), and forms that use the macro-replacement facilities.

An absolute expression must be a single term that is either a length-attribute reference, or a self-defining constant.

Use care with equated symbols, which should be used only when referring to registers (pointer references). For example, if an equated symbol is used for a length, it is treated as the address of the length and an unpredictable error occurs.

For non-Language Environment® (LE) AMODE(64) assembler language programs, argument values can be replaced as follows:

- *data-value* can be replaced by a relocatable expression that is an assembler-language reference to data of the correct type for the argument, or by a constant of the correct type for the argument.
- *data-area* can be replaced by a relocatable expression that is an assembler-language reference to data of the correct type for the argument.
- *data-area64* can be replaced by a relocatable expression that is an assembler-language 64-bit reference to data of the correct type for the argument.
- *cvda* is described in CICS-value data areas (CVDAs).
- *ptr-value* can be replaced by an absolute expression that is an assembler-language reference to a register.
- *ptr-value64* can be replaced by an absolute expression that is an assembler-language 64-bit reference to a register.
- *ptr-ref* can be replaced by an absolute expression that is an assembler-language reference to a register.
- *ptr-ref64* can be replaced by an absolute expression that is an assembler-language 64-bit reference to a register.
- *name* can be replaced either by a character string in single quotation marks, or by an assembler-language language relocatable expression reference to a character string. The length is equal to the maximum length allowed for the name. The value of the character string is the name to be used by the argument.

  filename, as used in FILE(*filename)*, specifies the name of the file. The name must have 1–8 characters from the range A–Z, 0–9, $, @, and #.

  systemname, as used in SYSID(*systemname*), specifies the name of the system to which the request is directed. The name must have 1–4 characters from the range A–Z, 0–9, $, @, and #.
- *hhmmss* can be replaced by a self-defining decimal constant, or an assembler-language reference to a field defined as PL4. The value must be of the form 0HHMMSS+ where:

  HH Represents hours from 00 through 99

  MM Represents minutes from 00 through 59

  SS Represents seconds from 00 through 59

label is not supported for AMODE(64) programs.

Many commands involve the transfer of data between the application program and CICS. In most cases, the application program must provide the length of the data to be transferred. However, if a data area is specified as the source or target, it is not necessary to provide the length explicitly, because the command-language

translator generates a default length. For example:

```
xxx DC CL8
 .
 .
EXEC CICS ... LENGTH(L'xxx)
```

CICS command restrictions

Some general restrictions apply to all CICS commands that access user data.

- The program must be in primary addressing mode when invoking any CICS service. The primary address space must be the home address space. All parameters passed to CICS must reside in the primary address space.
- If your program uses access registers, CICS preserves only access registers 2 through 13, because CICS code can use access registers 0, 1, 14 and 15 for z/OS® macro calls.

LENGTH options in CICS commands

In COBOL, PL/I, and Assembler language, the translator defaults certain lengths, if the NOLENGTH translator option is not specified. This means they are optional in programs that specify data areas. In C, all LENGTH options must be specified.

When a CICS command offers the LENGTH option, it is expressed as a signed halfword binary value. This puts a theoretical upper limit of 32 763 bytes on LENGTH. In practice, depending on issues of recoverability, function shipping, and other factors, assume a 24 KB limit.

This advisory 24 KB limit does not apply to the FLENGTH option on CICS commands (except for terminal-related **SEND** and **RECEIVE** commands, because of architectural limitations). The FLENGTH option is used on commands that relate to containers and journals, among others.

For temporary storage, transient data, and file control commands, the data set definitions themselves might impose further restrictions.

NOHANDLE option

Use the NOHANDLE option with any command to specify that you want no action to be taken for any condition or AID resulting from the execution of that command.

For further information about the NOHANDLE option, see RESP and NOHANDLE options .

Using the C or C++ language implies NOHANDLE on all commands.

RESP and RESP2 options

You can use the RESP option with any command to test whether a condition was raised during its execution. With some commands, when a condition can be raised for more than one reason, if you have already specified RESP, you can use the RESP2 option to determine exactly why a condition occurred.

RESP(*xxx*)
xxx is a user-defined fullword binary data area. On return from the command, it contains a value that corresponds to the condition that might be raised, or to a normal return, that is, xxx=DFHRESP(NORMAL). You can test this value by means of DFHRESP, as follows:

```
EXEC CICS WRITEQ TS FROM(abc)
                    QUEUE(qname)
                    NOSUSPEND
                    RESP(xxx)
                    RESP2(yyy)
   .
   .
IF xxx=DFHRESP(NOSPACE) THEN ...
```

This form of DFHRESP applies to both COBOL and PL/I.

The following example is a similar test in C:

```
switch (xxx) {
  case DFHRESP(NORMAL) : break;
  case DFHRESP(INVREQ) : Invreq_Cond();
                         break;
  default              : Errors();
}
```

The following example is a similar test in assembler language:

```
CLC   xxx,DFHRESP(NOSPACE)
```

The translator changes this code to:

```
CLC   xxx,=F'18'
```

Because the use of RESP implies NOHANDLE, use care when you use RESP with the RECEIVE command. NOHANDLE overrides both the HANDLE AID and the HANDLE CONDITION command, with the result that PF key responses are ignored.

RESP2(*yyy*)
yyy is a user-defined fullword binary data area. On return from the command, it contains a value that further qualifies the response to certain commands. Unlike the RESP values, RESP2 values have no associated symbolic names and there is no translator built-in function that corresponds to DFHRESP, so you must test the fullword binary value itself.

Translated code for CICS commands

Application programs can be written in COBOL, C, PL/I, or assembler language, and contain CICS commands. CICS translates these programs and creates an equivalent source program where each command is now translated into a call macro or statement in the language of the original source program.

COBOL translation output

EXEC CICS commands are converted to calls to the CICS interface DFHEI1.

The following example shows how the EXEC statement:
is translated to:

```
EXEC CICS RETURN TRANSID('fred')
     COMMAREA(mycommarea) END-EXEC.
```

```
Move length of mycommarea to dfhb0020
Call 'DFHEI1' using by content
          x'0e08e0000700001000f0f0f0f2f7404040'
          by content 'fred' by reference mycommarea
          by reference dfhb0020 end-call.
```

Copybook DFHEIBLC

This new copybook is a lower case version of the existing DFHEIBLK copybook.

A difference is that in DFHEIBLK the top level name is

```
01 EIBLK.
```

whereas in DFHEIBLC the top level name is

```
01 dfheiblk.
```

This is consistent with the name generated by the translator today, and also conforms to the rule that CICS reserved words should start with DFH.

C translation output

For a C application program, each command is replaced by reassignment statements followed by a dfhexec statement that passes the parameters.

PL/I translation output

For a PL/I application program, each command is always replaced by a DO statement, a declaration of a generated entry name, a CALL statement, and an END statement. The ENTRY declaration ensures that the appropriate conversions for argument values take place.

If a PL/I on-unit consists of a single EXEC CICS command, the command should be inside a BEGIN block, for example:

```
ON ERROR BEGIN;
         EXEC CICS RETURN;
         END;
```

In a similar way, if an EXEC CICS command is associated with a PL/I condition prefix, the command should be inside a BEGIN block, for example:

```
(NOZERODIVIDE): BEGIN;
                EXEC CICS GETMAIN
                SET(ptr-ref)
                LENGTH(data-value);
                END;
```

If OPTIONS(MAIN) is specified, the translator modifies the parameter list by inserting the EIB structure pointer as the first parameter. If OPTIONS(MAIN) is not specified (that is, if the program is to be link-edited to the main module), the parameter list is not modified, and it is the application programmer's responsibility to address the EIB structure in the link-edited program if access to it is required. In either case, where a program commences with a valid PL/I PROCEDURE statement, the translator inserts the declaration of the EIB structure.

Assembler translation output

The invocation of a CICS assembler language application program obeys system standards.

On entry to the application program, registers 1, 15, 14, and 13 contain the following addresses:

- Register 1 contains the address of the parameter list. This list has at least two entries:
  - Address of the EIB (EXEC interface block)
  - Address of the COMMAREA; if no COMMAREA, entry is X'00000000'
- Register 15 contains the address of the entry point.
- Register 14 contains the address of the return point.
- Register 13 contains the address of the save area.

All other registers are undefined.

DFHECALL macro

For an assembler language application program, when the CICS translator detects a CICS command, each command is replaced by an invocation of the DFHECALL macro. The DFHECALL macro sets up the command parameters and calls the initial CICS command processor to handle the command.

This macro expands to a system-standard call sequence that uses registers 15, 14, 0, and 1. The contents of these registers are as follows:

- Register 15 contains the address of the entry point in the EXEC interface program.
- Register 14 contains the address of the return point in your application program.
- Register 0 is undefined.
- Register 1 contains the address of the parameter list.

The entry point held in register 15 is resolved in the EXEC interface processor that must be link-edited with your application program. For AMODE(24) and AMODE(31) applications, this EXEC interface processor is DFHEAI; for AMODE(64) applications, it is DFHEAG.

You can specify the exit from the application program by an EXEC CICS RETURN command in your source program. Alternatively, you can use the DFHEIRET macro, which restores the registers and returns control to the address in register 14. The translator inserts the DFHEIRET macro, with no parameters specified, immediately before the END statement, unless you specify the NOEPILOG translator option. You can use this macro to return from a top-level program, but is not advisable from a lower-level program.

During assembly, the DFHECALL macro builds an argument list in dynamic storage, so that the application program is reentrant. Then the macro invokes the EXEC interface program DFHEIP for AMODE(24) or AMODE(31) applications, or DFHEIG for AMODE(64) applications. These programs also obey system standards, as previously described.

For AMODE(64) applications, although the application and the initial command processor run in 64-bit addressing mode, the parameters that the DFHECALL macro sets up and passes to the initial command processor contain 31-bit addresses. Therefore, the storage in which the call parameters are built, the DFHEISTG storage, must be 31-bit storage (above 16 MB but below 2 GB).

In addition to the invocation of the DFHECALL macro, the translator also inserts the following macros into your source program:

DFHEIGBL

This macro sets globals if you are using EXEC DLI in either a batch or an online CICS application program. Within DFHEIGBL, if DFHEIDL is set to 1, this means that the program contains EXEC DLI commands. If DFHEIDB is set to 1, this means that the program is batch DL/I. If you are not using DL/I, it is commented and set to 0.

DFHEIENT

This macro is inserted after the first CSECT or START instruction. It performs prolog code to allocate working storage to hold any user variables and for CICS use:

- It saves registers
- It gets an initial allocation of the storage that is defined by DFHEISTG
- It sets up a base register (default register 3)
- It sets up a dynamic storage register (default register 13)
- It sets up a register to address the EIB (default register 11)

DFHEIRET

This macro performs epilog code to release the working storage of the application program:

- It restores registers.

  DFHEIRET RCREG=nn, where *nn* (any register number other than 13) contains the return code to be placed in register 15 after the registers are restored.
- It returns control to the address in register 14.

DFHEISTG and DFHEIEND

These macros define dynamic storage:

- They define the storage required for the parameter list
- They define a save area.

For further details about these macros with AMODE(64) applications, see Coding the EXEC CICS(r) assembler interface.

A copybook, DFHEIBLK, that contains a DSECT that describes the EIB, is also included automatically.

The program must have an END statement because the translator does not otherwise insert the default macros. Also CSECT or START and END must be in uppercase for the translator to recognize them.

The example in Figure 1 shows a simple assembler language application program that uses the BMS command SEND MAP to send a map to a terminal, followed by the output after program INSTRUCT is translated.

```
Source program

INSTRUCT CSECT
         EXEC CICS SEND MAP('DFH$AGA') MAPONLY ERASE
         END


This source program is translated to:

         DFHEIGBL ,                INSERTED BY TRANSLATOR
INSTRUCT CSECT
         DFHEIENT                  INSERTED BY TRANSLATOR
*        EXEC CICS SEND MAP('DFH$AGA') MAPONLY ERASE
         DFHECALL =X'1804C000080000000046204000020',
               (CHA7,=CL7'DFH$AGA*'),(______RF,DFHEIV00)
         DFHEIRET                  INSERTED BY TRANSLATOR
         DFHEISTG                  INSERTED BY TRANSLATOR
         DFHEIEND                  INSERTED BY TRANSLATOR
         END
```

Figure 1. Source program and translated code for a CICS command

Chapter 2. CICS command summary

The **EXEC CICS** commands categorized according to the function they perform.

Abend support

- ABEND
- HANDLE ABEND

APPC basic conversation

- GDS ALLOCATE
- GDS ASSIGN
- GDS CONNECT PROCESS
- GDS EXTRACT ATTRIBUTES
- GDS EXTRACT PROCESS
- GDS FREE
- GDS ISSUE ABEND
- GDS ISSUE CONFIRMATION
- GDS ISSUE ERROR
- GDS ISSUE PREPARE
- GDS ISSUE SIGNAL
- GDS RECEIVE
- GDS SEND
- GDS WAIT

APPC mapped conversation

- ALLOCATE (APPC)
- CONNECT PROCESS
- CONVERSE (APPC)
- EXTRACT ATTRIBUTES (APPC)
- EXTRACT PROCESS
- FREE (APPC)
- ISSUE ABEND
- ISSUE CONFIRMATION
- ISSUE ERROR
- ISSUE PREPARE
- ISSUE SIGNAL (APPC)
- RECEIVE (APPC)
- SEND (APPC)
- WAIT CONVID (APPC)

Asynchronous Services

- FETCH ANY
- FETCH CHILD
- RUN TRANSID

© Copyright IBM Corp. 1989, 2017

Authentication

- CHANGE PASSWORD
- "CHANGE PHRASE" on page 70
- SIGNOFF
- SIGNON
- SIGNON TOKEN
- VERIFY PASSWORD
- VERIFY PHRASE
- VERIFY TOKEN

Batch data interchange

- ISSUE ABORT
- ISSUE ADD
- ISSUE END
- ISSUE ERASE
- ISSUE NOTE
- ISSUE QUERY
- ISSUE RECEIVE
- ISSUE REPLACE
- ISSUE SEND
- ISSUE WAIT

BMS

- PURGE MESSAGE
- RECEIVE MAP
- RECEIVE MAP MAPPINGDEV
- RECEIVE PARTN
- ROUTE
- SEND CONTROL
- SEND MAP
- SEND MAP MAPPINGDEV
- SEND PAGE
- SEND PARTNSET
- SEND TEXT
- SEND TEXT MAPPED
- SEND TEXTNOEDIT

Built-in functions

- BIF DEEDIT
- BIF DIGEST

CICS business transaction services (BTS)

- ACQUIRE
- ADD SUBEVENT
- CANCEL
- "CANCEL (BTS)" on page 68

- CHECK ACQPROCESS
- CHECK ACTIVITY
- CHECK TIMER
- DEFINE ACTIVITY
- DEFINE COMPOSITE EVENT
- DEFINE INPUT EVENT
- DEFINE PROCESS
- DEFINE TIMER
- DELETE ACTIVITY
- DELETE CONTAINER (BTS)
- DELETE EVENT
- DELETE TIMER
- ENDBROWSE ACTIVITY
- "ENDBROWSE CONTAINER (BTS)" on page 161
- ENDBROWSE EVENT
- ENDBROWSE PROCESS
- FORCE TIMER
- GET CONTAINER (BTS)
- GETNEXT ACTIVITY
- "GETNEXT CONTAINER (BTS)" on page 253
- GETNEXT EVENT
- GETNEXT PROCESS
- INQUIRE ACTIVITYID
- INQUIRE CONTAINER
- INQUIRE EVENT
- INQUIRE PROCESS
- INQUIRE TIMER
- LINK ACQPROCESS
- LINK ACTIVITY
- MOVE CONTAINER(BTS)
- PUT CONTAINER (BTS)
- REMOVE SUBEVENT
- RESET ACQPROCESS
- RESET ACTIVITY
- RESUME
- RETRIEVE REATTACH EVENT
- RETRIEVE SUBEVENT
- RUN
- STARTBROWSE ACTIVITY
- "STARTBROWSE CONTAINER (BTS)" on page 588
- STARTBROWSE EVENT
- STARTBROWSE PROCESS
- SUSPEND (BTS)
- TEST EVENT

Channel commands

- DELETE CHANNEL
- DELETE CONTAINER (CHANNEL)
- "ENDBROWSE CONTAINER (CHANNEL)" on page 161
- GET CONTAINER (CHANNEL)
- GET64 CONTAINER
- "GETNEXT CONTAINER (CHANNEL)" on page 254
- MOVE CONTAINER (CHANNEL)
- PUT CONTAINER (CHANNEL)
- PUT64 CONTAINER
- QUERY CHANNEL
- "STARTBROWSE CONTAINER (CHANNEL)" on page 590
- START CHANNEL

Console support

- WRITE OPERATOR

Diagnostic services

- DUMP TRANSACTION
- ENTER TRACENUM

Document services

- DOCUMENT CREATE
- DOCUMENT DELETE
- DOCUMENT INSERT
- DOCUMENT RETRIEVE
- DOCUMENT SET

Environment services

- ADDRESS
- ADDRESS SET
- ASSIGN

Event processing

- SIGNAL EVENT

Exception support

- HANDLE CONDITION
- IGNORE CONDITION
- POP HANDLE
- PUSH HANDLE

File control services

- DELETE
- ENDBR
- READ
- READNEXT
- READPREV

- RESETBR
- REWRITE
- STARTBR
- UNLOCK
- WRITE

Interval control services

- ASKTIME
- CANCEL
- DELAY
- FORMATTIME
- POST
- RETRIEVE
- START
- WAIT EVENT

Journaling

- WAIT JOURNALNAME
- WAIT JOURNALNUM
- WRITE JOURNALNAME
- WRITE JOURNALNUM

Monitoring

- MONITOR

Named counter server

- DEFINE COUNTER and DEFINE DCOUNTER
- DELETE COUNTER and DELETE DCOUNTER
- GET COUNTER and GET DCOUNTER
- QUERY COUNTER and QUERY DCOUNTER
- REWIND COUNTER and REWIND DCOUNTER
- UPDATE COUNTER and UPDATE DCOUNTER

Program control

- "INVOKE APPLICATION" on page 278
- LINK
- LOAD
- RELEASE
- RETURN
- XCTL

Scheduling services

- START ATTACH
- START BREXIT

Security services

- REQUEST PASSTICKET
- QUERY SECURITY

Spool Interface (JES)

- SPOOLCLOSE
- SPOOLOPEN INPUT
- SPOOLOPEN OUTPUT
- SPOOLREAD
- SPOOLWRITE

Storage control

- FREEMAIN
- FREEMAIN64
- GETMAIN
- GETMAIN64

Syncpoint

- SYNCPOINT
- SYNCPOINT ROLLBACK

Task control

- CHANGE TASK
- DEQ
- ENQ
- SUSPEND
- WAIT EXTERNAL
- WAITCICS

TCP/IP services

- EXTRACT CERTIFICATE
- EXTRACT TCPIP

Temporary storage control

- DELETEQ TS
- READQ TS
- WRITEQ TS

Terminal control

- ALLOCATE (LUTYPE6.1)
- ALLOCATE (MRO)
- BUILD ATTACH (LUTYPE6.1)
- BUILD ATTACH (MRO)
- CONVERSE (APPC)
- "CONVERSE (default)" on page 83
- CONVERSE (LUTYPE2/LUTYPE3)
- CONVERSE (LUTYPE4)
- CONVERSE (LUTYPE6.1)
- CONVERSE (MRO)
- "CONVERSE (non-z/OS Communications Server default)" on page 96
- "CONVERSE: non-z/OS Communications Server options" on page 97

- CONVERSE (SCS)
- "CONVERSE: z/OS Communications Server options" on page 91
- CONVERSE (2260)
- CONVERSE (3270 logical)
- CONVERSE (3600-3601)
- CONVERSE (3600-3614)
- CONVERSE (3650 interpreter)
- CONVERSE (3650-3270)
- CONVERSE (3650-3653)
- CONVERSE (3650-3680)
- CONVERSE (3767)
- CONVERSE (3770)
- CONVERSE (3790 full-function or inquiry)
- CONVERSE (3790 3270-display)
- EXTRACT ATTACH (LUTYPE6.1)
- EXTRACT ATTACH (MRO)
- EXTRACT ATTRIBUTES (MRO)
- EXTRACT LOGONMSG
- EXTRACT TCT
- FREE (LUTYPE6.1)
- FREE
- FREE (MRO)
- HANDLE AID
- ISSUE COPY (3270 logical)
- "ISSUE DISCONNECT (LUTYPE6.1)" on page 296
- ISSUE DISCONNECT
- ISSUE ENDFILE
- ISSUE ENDOUTPUT
- ISSUE EODS
- ISSUE ERASEAUP
- ISSUE LOAD
- ISSUE PASS
- ISSUE PRINT
- ISSUE RESET
- ISSUE SIGNAL (LUTYPE6.1)
- POINT
- RECEIVE (APPC)
- RECEIVE (LUTYPE2/LUTYPE3)
- RECEIVE (LUTYPE4)
- RECEIVE (LUTYPE6.1)
- RECEIVE (MRO)
- "RECEIVE (non-z/OS Communications Server default)" on page 422
- "RECEIVE: non-z/OSCommunications Server options" on page 426
- "RECEIVE (z/OS Communications Server default)" on page 412
- "RECEIVE (z/OS Communications Server default)" on page 412

- RECEIVE (2260)
- RECEIVE (2980)
- RECEIVE (3270 logical)
- "RECEIVE (3600 pipeline)" on page 415
- RECEIVE (3600-3601)
- RECEIVE (3600-3614)
- RECEIVE (3650)
- RECEIVE (3767)
- RECEIVE (3770)
- RECEIVE (3790 full-function or inquiry)
- RECEIVE (3790 3270-display)
- SEND (APPC)
- SEND (LUTYPE2/LUTYPE3)
- SEND (LUTYPE4)
- SEND (LUTYPE6.1)
- SEND (MRO)
- "SEND (non-z/OS Communications Server default)" on page 494
- "SEND: non-z/OS Communications Server options" on page 496
- SEND (SCS)
- "SEND (z/OS Communications Server default)" on page 481
- "SEND: z/OS Communications Server options" on page 490
- SEND (2260)
- SEND (2980)
- SEND (3270 logical)
- SEND (3600 pipeline)
- SEND (3600-3601)
- SEND (3600-3614)
- SEND (3650 interpreter)
- SEND (3650-3270)
- SEND (3650-3653)
- SEND (3650-3680)
- SEND (3767)
- SEND (3770)
- SEND (3790 full-function or inquiry)
- SEND (3790 SCS)
- SEND (3790 3270-display)
- SEND (3790 3270-printer)
- WAIT SIGNAL
- WAIT TERMINAL

Transient data

- DELETEQ TD
- READQ TD
- WRITEQ TD

Web support

- CONVERTTIME
- EXTRACT WEB
- WEB CLOSE
- WEB CONVERSE
- WEB ENDBROWSE FORMFIELD
- WEB ENDBROWSE HTTPHEADER
- "WEB ENDBROWSE QUERYPARM" on page 651
- WEB EXTRACT
- WEB OPEN
- WEB PARSE URL
- WEB READ FORMFIELD
- WEB READ HTTPHEADER
- "WEB READ QUERYPARM" on page 668
- WEB READNEXT FORMFIELD
- WEB READNEXT HTTPHEADER
- "WEB READNEXT QUERYPARM" on page 673
- WEB RECEIVE (Server)
- WEB RECEIVE (Client)
- WEB RETRIEVE
- WEB SEND (Server)
- WEB SEND (Client)
- WEB STARTBROWSE FORMFIELD
- WEB STARTBROWSE HTTPHEADER
- "WEB STARTBROWSE QUERYPARM" on page 711
- WEB WRITE HTTPHEADER

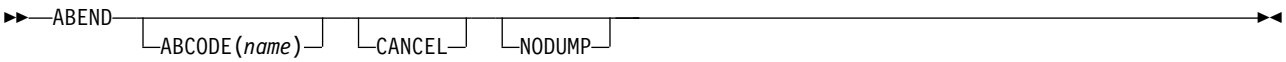
Web services

- INVOKE SERVICE
- INVOKE WEBSERVICE
- SOAPFAULT ADD
- SOAPFAULT CREATE
- SOAPFAULT DELETE
- "TRANSFORM DATATOJSON" on page 597
- TRANSFORM DATATOXML
- "TRANSFORM JSONTODATA" on page 602
- TRANSFORM XMLTODATA
- WSACONTEXT BUILD
- WSACONTEXT DELETE
- WSACONTEXT GET
- WSAEPR CREATE

ABEND

Terminate a task abnormally.

ABEND



This command is threadsafe.

Description

The **ABEND** command terminates a task abnormally.

CICS releases the main storage associated with the terminated task; optionally, you can obtain a transaction dump of this storage.

Invoking the **ABEND** command causes the current transaction to abend. The Language Environment is informed that an abend has occurred and the following message is written out to CEEMSG followed by a dump report:

```
CEE3250C The system or user abend XXXX was issued
```

XXXX is the transaction dump code specified on the ABCODE option. The Language Environment attempts to access register addresses to dump out the referenced storage as part of the dump report written to CEEMSG. If the Language Environment does not have access to the storage addressed by these registers, an 0C4 abend can occur. You can eliminate 0C4 abends by setting the Language Environment runtime option TERMTHDACT to QUIET, MSG, or UAONLY. For more information, see TERMTHDACT.

Options

ABCODE(*name***)**
Specifies that main storage related to the terminating task is dumped. The ABCODE is used as a transaction dump code to identify the dump. ABCODE follows the format rules for DUMPCODE. The “DUMP TRANSACTION” on page 153 command gives the format rules that apply to DUMPCODE, if these rules are not followed, ABEND does not produce a dump.

Do not start the name with the letter A, because this is reserved for CICS itself.

Note: If ABCODE is not used, the effect is the same as NODUMP.

CANCEL
Specifies that exits established by HANDLE ABEND commands are ignored. An **ABEND CANCEL** command cancels all exits at any level in the task and terminates the task abnormally. If the PL/I STAE execution-time option is specified, an abnormal termination exit is established by PL/I. This exit is revoked by the CANCEL option.

NODUMP
Specifies that an abend occurs without causing a dump to be taken. For programs link-edited using the Language Environment SCEELKED library, when NODUMP is specified, a dump is never taken, regardless of any setting in the transaction dump table. For programs not link-edited with Language Environment, if the transaction dump table already has an entry for the abend code, or if the abend is in Language Environment run-unit initialization or termination, the NODUMP option is ignored.

Examples

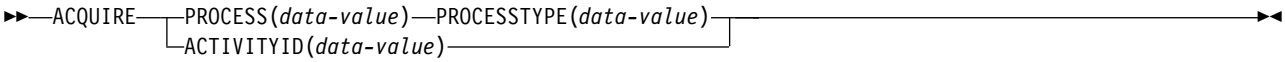
The following example shows how to terminate a task abnormally:

```
EXEC CICS ABEND ABCODE('BCDE')
```

ACQUIRE

Acquire access to a BTS activity from outside the process that contains it.

ACQUIRE PROCESS



Conditions: ACTIVITYBUSY, ACTIVITYERR, INVREQ, IOERR, LOCKED, NOTAUTH, PROCESSBUSY, PROCESSERR

Description

ACQUIRE enables a program that is executing outside a particular BTS process to access an activity within the process. It allows the program to:

- Read and write to the activity's data-containers
- Issue various commands, such as RUN and LINK, against the activity.[1]

An activity that a program gains access to by means of an ACQUIRE command is known as an **acquired activity**. A program can acquire only one activity per unit of work. The activity remains acquired until the next syncpoint.

ACQUIRE ACTIVITYID acquires the specified descendant (non-root) activity.

ACQUIRE PROCESS acquires the root activity of the specified process.

Note: When a program defines a process, it is automatically given access to the process's root activity. (This enables the defining program to access the process containers and root activity containers before running the process.) When a program gains access to a root activity by means of *either* a DEFINE PROCESS or an ACQUIRE PROCESS command, the process is known as the **acquired process**.

Rules

1. A program can acquire only one activity within the same unit of work. The activity remains acquired until the next syncpoint. This means, for example, that a program:
   - Cannot issue both a DEFINE PROCESS and an ACQUIRE PROCESS command within the same unit of work.
   - Cannot issue both an ACQUIRE PROCESS and an ACQUIRE ACTIVITYID command within the same unit of work. That is, it can acquire *either* a descendant activity or a root activity, not one of each.
2. If a program is executing as an activation of an activity, it cannot:
   - Acquire an activity in the same process as itself. It cannot, for example, issue ACQUIRE PROCESS for the current process.

1. If the acquired activity is a root activity, against the process.

- Use a LINK command to activate the activity that it has acquired.

3. An acquired activity's process is accessible in the same way as the activity itself can access it. Thus, if the acquired activity is a descendant activity:
   - Its process's containers may be read but not updated.
   - The process may not be the subject of any command—such as RUN, LINK, SUSPEND, RESUME, or RESET—that directly manipulates the process or its root activity.

   Conversely, if the acquired activity is a root activity:
   - Its process's containers may be both read and updated.
   - The process may be the subject of commands such as RUN, LINK, SUSPEND, RESUME, or RESET. The ACQPROCESS keyword on the command identifies the subject process as the one the program that issues the command has acquired in the current unit of work.

Options

ACTIVITYID(data-value)
specifies the identifier (1–52 characters) of the descendant activity to be acquired.

PROCESS(data-value)
specifies the name (1–36 characters) of the process whose root activity is to be acquired.

PROCESSTYPE(data-value)
specifies the process-type (1–8 characters) of the process whose root activity is to be acquired.

Conditions

107 ACTIVITYBUSY
RESP2 values:

19 The request timed out. It may be that another task using this activity-record has been prevented from ending.

109 ACTIVITYERR
RESP2 values:

8 The activity referred to by the ACTIVITYID option could not be found.

16 INVREQ
RESP2 values:

22 The unit of work that issued the ACQUIRE command has already acquired an activity; a unit of work can acquire only one activity.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access the file associated with the BTS repository data set on which details of the process are stored.

106 PROCESSBUSY
RESP2 values:

13 The request timed out. It may be that another task using this process-record has been prevented from ending.

108 PROCESSERR
RESP2 values:

5 The process named in the PROCESS option could not be found.

9 The process-type named in the PROCESSTYPE option could not be found.

Usage examples

ACQUIRE ACTIVITYID can be used to implement user-related activities. For example, on its first activation an activity might:

1. Define an input event to represent a particular user-interaction
2. Issue an ASSIGN command to obtain the identifier of its own activity-instance
3. Save the input event and activity identifier on a data base
4. Return without completing.

Later, when a user is ready to process the work represented by the activity, he or she starts a transaction. This transaction, which executes outside the BTS process:

1. Retrieves the input event and activity identifier from the data base
2. Uses the ACQUIRE ACTIVITYID command to acquire access to the activity
3. Places the information required to complete the activity in an input data-container, and runs the activity. The INPUTEVENT option of the RUN command tells the activity why it is being activated.

ACQUIRE PROCESS can be used to implement client/server processing. For example, a client program might use the DEFINE PROCESS and RUN commands to create and run a server process, which carries out some work, defines one or more input events, and returns without completing. The client issues a syncpoint or returns. To run the same server process again, the client uses the ACQUIRE PROCESS and RUN commands.

ADD SUBEVENT

Add a sub-event to a BTS composite event.

ADD SUBEVENT



Conditions: EVENTERR, INVREQ

Description

ADD SUBEVENT adds a sub-event to a BTS composite event. The sub-event:

- Must be an atomic (not a composite) event
- Cannot be a system event
- Must not currently be part of a composite event
- Cannot, if the predicate of the composite event uses the AND Boolean operator, be an input event.

Adding a sub-event causes the composite's predicate to be re-evaluated.

Options

EVENT(data-value)
specifies the name (1–16 characters) of the composite event. This must previously have been defined to the current activity, using the DEFINE COMPOSITE EVENT command.

SUBEVENT(data-value)
specifies the name (1–16 characters) of the atomic event to be added to the composite event as a sub-event. The sub-event must previously have been defined to the current activity, using one of the following commands:

- DEFINE ACTIVITY
- DEFINE INPUT EVENT
- DEFINE TIMER

It:

- Must not currently be part of a composite event
- Cannot, if the predicate of the composite event uses the AND Boolean operator, be an input event.

Conditions

111 EVENTERR
RESP2 values:

4 The event specified on the EVENT option is not recognized by BTS.

5 The sub-event specified on the SUBEVENT option is not recognized by . BTS.

16 INVREQ
RESP2 values:

1 The command was issued outside the scope of an activity.

2 The event specified on the EVENT option is invalid—it is not a composite event.

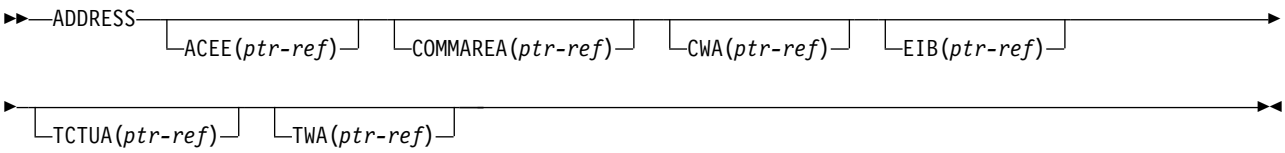
3 The sub-event specified on the SUBEVENT option is invalid. Specifying any of the following as a sub-event produces this error:

- A composite event
- A system event
- A sub-event of another composite event
- A sub-event of this composite event—that is, an atomic event that has already been added to this composite event
- An input event, if the composite uses the AND Boolean operator.

ADDRESS

Obtain access to CICS storage areas.

ADDRESS



This command is threadsafe.

Note for dynamic transaction routing: Using ADDRESS with CWA could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. See Affinity for more information about transaction affinities.

Description

ADDRESS accesses the following areas:

- The access control environment element (ACEE)
- The communication area available to the invoked program (COMMAREA)
- The common work area (CWA)
- The EXEC interface block (EIB)
- The terminal control table user area (TCTUA)
- The transaction work area (TWA)

In assembler language, no more than four options may be specified in one ADDRESS command.

Options

ACEE(*ptr-ref***)**
returns a pointer to the access control environment element, the control block that is generated by an external security manager (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER's ACEE is returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

For information on how to map the ACEE data area, see the mapping macro IHAACEE supplied in SYS1.MACLIB.

Note: Take care when addressing an ACEE in a server program invoked by a distributed program link. The ACEE address returned depends on the link security and may not be the same as the address of the user signed on at the local system.

COMMAREA(*ptr-ref***)**
returns a pointer reference, set to the address of the communication area (COMMAREA) available to the currently executing program. COMMAREA is used to pass information between application programs. If the COMMAREA does not exist, the pointer reference is set to the null value, X'FF000000'.

In C, you must use ADDRESS COMMAREA to get the address of the communication area, because this is not passed as an argument to a C main function.

CWA(*ptr-ref***)**
returns a pointer reference, set to the address of the common work area (CWA). This area makes information available to applications running in a single CICS system. If a CWA does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

EIB(*ptr-ref***)**
returns a pointer reference set to the address of the EXEC interface block (EIB). You must use this option to get addressability to the EIB in application routines other than the first invoked by CICS (where addressability to the EIB is provided automatically). If the application program is translated with SYSEIB in the XOPTS parameter list, this option returns the address of the system EIB.

If TASKDATALOC(ANY) is defined on the transaction definition, the address of the data may be above or below the 16MB line.

If TASKDATALOC(BELOW) is defined on the transaction definition, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

C functions must use ADDRESS EIB to get the address of the EXEC interface block, because this address is not passed as an argument to a C main function. You must code an ADDRESS EIB statement at the beginning of each application if you want access to the EIB, or if you are using a command that includes the RESP or RESP2 option.

TCTUA(*ptr-ref***)**
returns a pointer reference, set to the address of the terminal control table user area (TCTUA) for the principal facility, not that for any alternate facility that may have been allocated. This area is used for passing information between application programs, but only if the same terminal is associated with the application programs involved. If a TCTUA does not exist, the pointer reference is set to the null value, X'FF000000'.

TWA(*ptr-ref***)**
returns a pointer reference, set to the address of the transaction work area (TWA). This area is used for passing information between application programs, but only if they are in the same task. If a TWA does not exist, the pointer reference is set to the null value, X'FF000000'.

If TASKDATALOC(ANY) is defined on the transaction definition, the address of the data may be above or below the 16MB line.

If TASKDATALOC(BELOW) is defined on the transaction definition, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

ADDRESS SET

Set the address of a structure or pointer.

ADDRESS SET



Description

The value from the USING option is used to set the reference in the SET option.

Options

SET(*data-area/ptr-ref***)**
sets a pointer reference.

USING(*ptr-ref/data-area***)**
supplies a pointer value.

COBOL example of ADDRESS SET

To set the address of a structure to a known pointer value:

```
EXEC CICS ADDRESS SET(address of struc)
          USING(ptr)
```

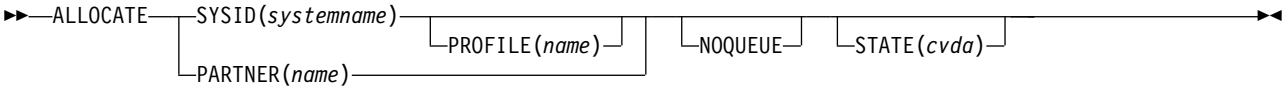
To set a pointer variable to the address of a structure:

```
EXEC CICS ADDRESS SET(ptr)
          USING(address of struc01)
```

ALLOCATE (APPC)

Allocate a session to a remote APPC logical unit for use by an APPC mapped conversation.

ALLOCATE (APPC)



Conditions: CBIDERR, INVREQ, NETNAMEIDERR, PARTNERIDERR, SYSBUSY, SYSIDERR

Description

ALLOCATE makes one of the sessions associated with the named system available to the application program, and optionally selects a set of session-processing options.

CICS returns, in EIBRSRCE in the EIB, the 4-byte CONVID (conversation identifier) that the application program uses in all subsequent commands that relate to the conversation.

If a session to the requested APPC LU is not available, the application is suspended until a session does become available. In such a case, the suspension of the application can be prevented by specifying either the NOQUEUE or the NOSUSPEND option. NOSUSPEND is still supported as an equivalent for NOQUEUE, but NOQUEUE is the preferred keyword.

A session is immediately available for allocation only if it is all of the following:

- A contention winner
- Already bound
- Not already allocated

CICS attempts to satisfy a request for a session by choosing among sessions in the following order of preference:

1. Contention winner that is bound and not already allocated (CICS allocates it). This is a session that is immediately available.
2. Contention winner that is unbound (CICS binds it and allocates it).
3. Contention loser that is bound and not already allocated (CICS bids for it).
4. Contention loser that is unbound (CICS binds it and bids for it).

The action taken by CICS if no session is immediately available depends on whether you specify NOQUEUE (or the equivalent NOSUSPEND option), and also on whether your application has executed a HANDLE command for the SYSBUSY condition. In these situations, CICS does not bid for sessions or bind additional sessions. It looks for a session that is immediately available (that is, a contention winner that is bound and not already allocated), and if one is not available, the SYSBUSY condition is returned. The possible combinations are shown below:

HANDLE for SYSBUSY condition issued
: The command is not queued and control is returned immediately to the label specified in the HANDLE command, whether or not you have specified NOQUEUE.

No HANDLE issued for SYSBUSY condition
: If you have specified NOQUEUE (or NOSUSPEND), the request is not queued and control is returned immediately to your application program. The SYSBUSY code (X'D3') is set in the EIBRCODE field of the EXEC interface block. You should test this field immediately after issuing the ALLOCATE command.

The HANDLE for SYSBUSY condition therefore has the same effect as the NOQUEUE option, except for where control is returned in the application. If the HANDLE command is used, control is returned to the label, and if it is not used, control is returned to the instruction following the ALLOCATE command.

If you have omitted the NOQUEUE option, and you have not issued a HANDLE command for the SYSBUSY option, then if no session is immediately available, CICS queues the request (and your application waits) until a session is available. The request is allocated a session either when a winner session has become available, or when CICS has successfully bid for a loser session. Omit the NOQUEUE option when you want all winner or loser sessions to be considered for allocation to the request. You can use the QUEUELIMIT and MAXQTIME attributes of the CONNECTION resource definitions to limit the length of the queue of requests, and the time that requests spend on the queue. Managing allocate queues has more information about allocate queues. The DTIMOUT value specified on the transaction definition can be used to limit the wait time for individual requests.

Options

NOQUEUE
overrides the default action when a SYSBUSY condition arises. This indicates that a session is not immediately available. The default action is to suspend application execution until a session is available. NOQUEUE inhibits this waiting; control returns immediately to the application program instruction following the command.

Note, however, that if a HANDLE CONDITION for SYSBUSY is active when the command is executed, this also overrides the default action, and control is passed to the user label supplied in the HANDLE CONDITION. This takes precedence over the NOQUEUE option but is, of course, negated by either NOHANDLE or RESP.

If an APPC ALLOCATE request is issued against a single session connection from the contention loser end, the NOQUEUE option always causes SYSBUSY to be returned rather than allowing the request to bid for the session. If the NOQUEUE option is absent, the request is able to bid for the session.

If an APPC ALLOCATE request is issued against a parallel session connection, and the NOQUEUE option is specified, only sessions that are immediately available (that is, a contention winner that is bound and not already allocated) can be allocated to the request. If no such session is available, then SYSBUSY is returned. If the NOQUEUE option is absent, the request is able to bid for a loser session, or bind unbound winner sessions.

PARTNER(*name*)
specifies the name (8 characters) of a set of definitions that include the names of a remote LU (NETNAME) and a communication profile to be used on the allocated session. You can use this option as an alternative to specifying SYSID and PROFILE explicitly.

PROFILE(*name*)
specifies the name (1-8 characters) of a set of session-processing options that are to be used during the execution of mapped commands for the session specified in the SYSID option. If you specify SYSID and omit PROFILE, a default profile (DFHCICSA) is selected.

STATE(*cvda*)
gets the state of the current conversation. The cvda value returned by CICS is ALLOCATED.

SYSID(*systemname*)
specifies the name (1-4 characters) by which the remote APPC LU is known to this CICS. This option requests that one of the sessions to the named system is to be allocated.

Conditions

62 CBIDERR
occurs if the requested PROFILE cannot be found.

Default action: terminate the task abnormally.

16 INVREQ
occurs if the ALLOCATE command is not valid for the device to which it is directed.

Default action: terminate the task abnormally.

99 NETNAMEIDERR
occurs if the name specified in the NETNAME parameter of the RDO definition for the PARTNER specified on the allocate command is invalid.

Default action: terminate the task abnormally.

97 PARTNERIDERR
occurs if the name specified in the PARTNER option is not recognized by CICS.

Default action: terminate the task abnormally.

59 SYSBUSY
occurs for one of the following reasons:

- The request for a session cannot be serviced immediately. This is only possible if the NOQUEUE option is set, or a HANDLE CONDITION for SYSBUSY is active.
- The ALLOCATE command is issued when persistent session recovery is still in process and the sessions needed to satisfy the command are not yet recovered.

Default action: ignore the condition.

53 SYSIDERR
occurs if CICS is unable to provide the application program with a suitable session, for one of the following reasons:

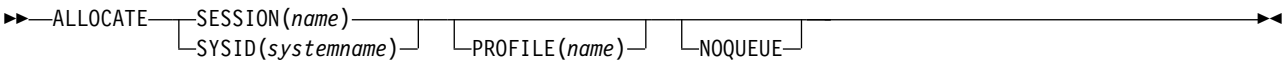
- The name specified in the SYSID option is not recognized by CICS.
- The mode name derived from the PROFILE option is not one of the mode names defined for the APPC system entry.
- All the sessions in the group specified by SYSID and mode name are out of service, or all sessions are out of service.
- The AID (automatic initiate descriptor) representing your ALLOCATE has been canceled.
- All the sessions are busy and the (queued) allocates have been purged or rejected.

Default action: terminate the task abnormally.

ALLOCATE (LUTYPE6.1)

Acquire a session to a remote LUTYPE6.1 logical unit.

ALLOCATE (LUTYPE6.1)



Conditions: CBIDERR, EOC, INVREQ, SESSBUSY, SESSIONERR, SYSBUSY, SYSIDERR

Description

ALLOCATE acquires an alternate facility and optionally selects a set of session-processing options. If SYSID is specified, CICS makes available to the application program one of the sessions associated with the named system. The name of this session can be obtained from EIBRSRCE in the EIB. If SESSION is specified, CICS makes the named session available.

If the session requested is not available, the application is suspended until the session does become available. In such a case, the suspension of the application can be prevented by specifying either the NOQUEUE or the NOSUSPEND option. NOSUSPEND is still supported as an equivalent for NOQUEUE, but NOQUEUE is the preferred keyword.

Options

NOQUEUE
overrides the default action when a SESSBUSY or SYSBUSY condition arises. These conditions indicate that the session requested is not immediately available. The default action is to suspend application execution until the session is available. NOQUEUE inhibits this waiting; control returns immediately to the application program instruction following the command.

Note, however, that if a HANDLE CONDITION for SESSBUSY or SYSBUSY is active when the command is executed, this also overrides the default action, and control is passed to the user label supplied in the HANDLE CONDITION. This takes precedence over the NOQUEUE option but is, of course, negated by either NOHANDLE or RESP.

PROFILE(*name*)
specifies the name (1–8 characters) of a set of session-processing options that are to be used during execution of terminal control commands for the session specified in the SYSID or SESSION options. If the PROFILE option is omitted, a default profile (DFHCICSA) is selected.

SESSION(*name*)
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used.

SYSID(*systemname*)
specifies the name (1–4 characters) of a system TCTSE. This option specifies that one of the sessions to the named system is to be allocated.

Conditions

62 CBIDERR
occurs if the requested PROFILE cannot be found.

Default action: terminate the task abnormally.

06 EOC
occurs when a request/response unit (RU) is received with the end-of-chain indicator set. Field EIBEOC also contains this indicator.

Default action: ignore the condition.

16 INVREQ
occurs when the specified session is already allocated to this task, or the session is an APPC session.

Default action: terminate the task abnormally.

60 SESSBUSY
occurs if the request for the specified session cannot be serviced immediately. This is only possible if the NOQUEUE option is set, or a HANDLE CONDITION for SESSBUSY is active.

Default action: ignore the condition.

58 SESSIONERR
occurs if the name specified in the SESSION option is not that of an LUTYPE6.1 session TCTTE, or if the session cannot be allocated because it is out of service.

Default action: terminate the task abnormally.

59 SYSBUSY
occurs for one of the following reasons:

- The request for a session cannot be serviced immediately. This is only possible if the NOQUEUE option is set, or a HANDLE CONDITION for SYSBUSY is active.
- The ALLOCATE command is issued when persistent session recovery is still in process and the sessions needed to satisfy the command are not yet recovered.

Default action: ignore the condition.

53 SYSIDERR
occurs if CICS is unable to provide the application program with a suitable session, for one of the following reasons:

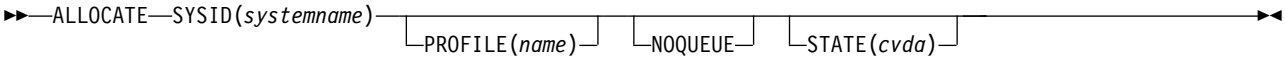
- The name specified in the SYSID option is not recognized by CICS.
- All sessions are out of service.
- The AID (automatic initiate descriptor) representing your ALLOCATE has been canceled.
- All the sessions are busy and the (queued) allocates have been purged or rejected.

Default action: terminate the task abnormally.

ALLOCATE (MRO)

Acquire an MRO session.

ALLOCATE (MRO)



Conditions: INVREQ, SYSBUSY, SYSIDERR

Description

ALLOCATE acquires an alternate facility. CICS makes available to the application program one of the sessions associated with the system named in the SYSID option. The name of this session can be obtained from EIBRSRCE in the EIB.

If the session requested is not available, the application is suspended until the session does become available. In such a case, the suspension of the application can be prevented by specifying the NOQUEUE option.

Options

NOQUEUE
overrides the default action when a SYSBUSY condition arises. This condition indicates that the session requested is not immediately available. The default action is to suspend application execution until the session is available. NOQUEUE inhibits this waiting; control returns immediately to the application program instruction following the command.

Note, however, that if a HANDLE CONDITION for SYSBUSY is active when the command is executed, this also overrides the default action, and control is passed to the user label supplied in the HANDLE CONDITION. This takes precedence over the NOQUEUE option but is, of course, negated by either NOHANDLE or RESP.

PROFILE(*name*)
specifies the name (1–8 characters) of a set of session-processing options that are to be used during execution of terminal control commands for the session specified in the SYSID option. If the PROFILE option is omitted, a default profile (DFHCICSA) is selected.

STATE(*cvda*)
gets the state of the current conversation. The cvda value returned by CICS is ALLOCATED.

SYSID(*systemname*)
specifies the name (1–4 characters) of a system TCTSE. This option specifies that one of the sessions to the named system is to be allocated.

Conditions

16 INVREQ
occurs if an incorrect command has been issued for the LU or terminal in use.

Default action: terminate the task abnormally.

59 SYSBUSY
occurs if the request for a session cannot be serviced immediately. This is only possible if the NOQUEUE option is set, or a HANDLE CONDITION for SYSBUSY is active.

Default action: ignore the condition.

53 SYSIDERR
occurs if CICS is unable to provide the application program with a suitable session, for one of the following reasons:

- The name specified in the SYSID option is not recognized by CICS.
- All sessions are out of service.
- The AID (automatic initiate descriptor) representing your ALLOCATE has been canceled.
- All the sessions are busy and the (queued) allocates have been purged or rejected.

Default action: terminate the task abnormally.

ASKTIME

Request current date and time of day.

ASKTIME



This command is threadsafe.

Description

ASKTIME updates the date (EIBDATE) and CICS time-of-day clock (EIBTIME) fields in the EIB. These two fields initially contain the date and time when the task started.

In response to an ASKTIME command, CICS issues an MVS™ STCK macro and modifies this by a local time difference. For example, if your MVS TOD (hardware) clock is set to GMT, and the local time is defined as British Summer Time (BST), it is BST that is stored in the EIBTIME field.

For details of the EIB, see EIB fields.

Options

ABSTIME(*data-area***)**
Specifies the data area for the number of milliseconds since 00:00 on 1 January 1900, which is known as absolute time. The time is taken from the system time-of-day clock, adjusted for leap seconds and to apply the local timezone offset (including daylight saving time), truncated to the millisecond, and returned as a packed decimal of length 8 bytes.

You can use FORMATTIME to change the data into other familiar formats.

Example

For example, after the following command runs:

```
EXEC CICS ASKTIME ABSTIME(utime)
```

utime contains a value similar in format to 002837962864820.

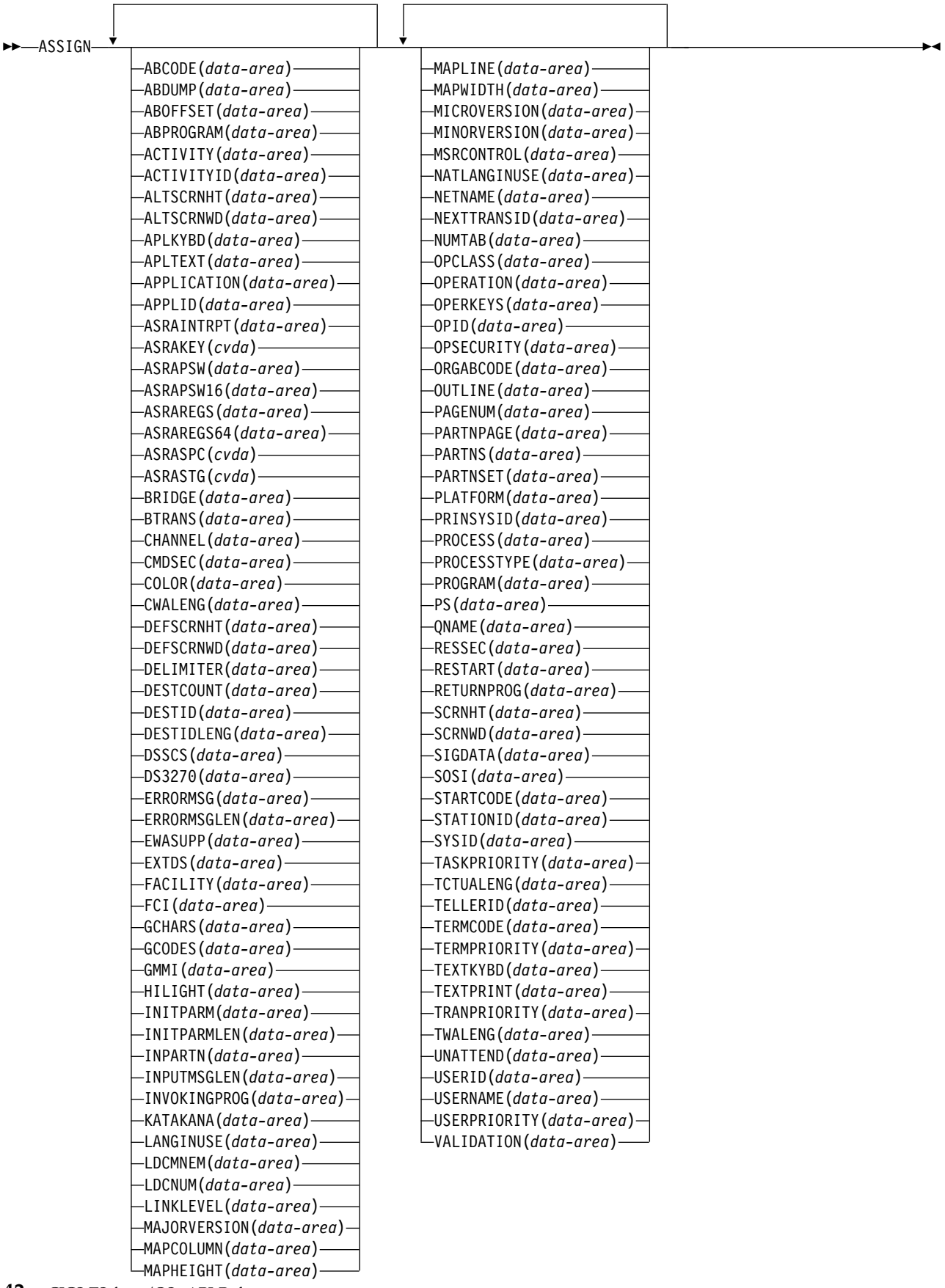
The format of *data-area* is:

```
COBOL:  PIC S9(15) COMP-3
C:      char data_area[8];
PL/I:   FIXED DEC(15)
ASM:    PL8
```

ASSIGN

Request values from outside the local environment of the application program.

ASSIGN



Condition: INVREQ

Description

The **ASSIGN** command gets values from outside the local environment of the application program. The data obtained depends on the specified options. You can specify up to 16 options in one **ASSIGN** command.

For options that apply to terminals or terminal-related data, the reference is always to the principal facility.

If the principal facility is a remote terminal, the data returned is obtained from the local copy of the information; the request is not routed to the system to which the remote terminal is attached.

Transaction routing is, as far as possible, transparent to the **ASSIGN** command. In general, the values returned are the same whether the transaction is local or remote.

For more details on these options, see Developing in an intersystem environment.

Options

ABCODE(*data-area***)**
Returns a 4-character current abend code. Abend codes are documented in Transaction abend codes. If an abend has not occurred, the variable is set to blanks.

ABDUMP(*data-area***)**
Returns a 1-byte value. `X'FF'` indicates that an **EXEC CICS ABEND ABCODE** command was issued without the `NODUMP` option and that ABCODE contains an abend code. `X'00'` indicates that either no dump was produced, or ABCODE contains blanks.

ABOFFSET(*data-area***)**
Returns a fullword binary offset in bytes of an abend when the latest abend with a code ASRA, ASRB, or ASRD occurred. If the abend is outside the current program, a value of `X'FFFFFFFF'` is returned. The data area is set to binary zeros if no ASRA, ASRB, or ASRD abend occurred during the execution of the issuing transaction, or if the abend originally occurred in a remote DPL server program.

ABPROGRAM(*data-area***)**
Returns an 8-character name of the failing program for the latest abend.

If the abend originally occurred in a DPL server program running in a remote system, ABPROGRAM returns the DPL server program name.

This field is set to binary zeros if it is not possible to determine the failing program at the time of the abend.

When the latest abend is an APCT (resulting from an unsuccessful attempt to load a program, mapset or partitionset), the name is taken from the program, mapset, or partitionset that was not loaded.

ACTIVITY(*data-area***)**
Returns, if this program is running on behalf of a CICS business transaction services (BTS) activity, the 16-character name of the activity.

BTS is described in Overview of BTS.

ACTIVITYID(*data-area***)**
Returns, if this program is running on behalf of a BTS activity, the 52-character, CICS-assigned, identifier of the activity-instance.

If a program that is running outside the current process wants to acquire control of this activity-instance, it must specify this identifier on an **ACQUIRE ACTIVITYID** command.

BTS is described in Overview of BTS.

ALTSCRNHT(*data-area***)**
Returns the alternate screen height defined for the terminal as a halfword binary variable. If the task is not initiated from a terminal, INVREQ occurs.

ALTSCRNWD(*data-area***)**
Returns the alternate screen width defined for the terminal as a halfword binary variable. If the task is not initiated from a terminal, INVREQ occurs.

APLKYBD(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal keyboard has the APL keyboard feature (`X'FF'`) or not (`X'00'`). If the task is not initiated from a terminal, INVREQ occurs.

APLTEXT(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal keyboard has the APL text feature (`X'FF'`) or not (`X'00'`). If the task is not initiated from a terminal, INVREQ occurs.

APPLICATION(*data-area***)**
Returns the 64 character name of the current application associated with the task. It is part of the application context that is made up of the application name, the platform name, the operation name and the major, minor and micro version number of the application. If there is no application context associated with the task, then blanks are returned.

APPLID(*data-area***)**
Returns an 8-character APPLID of the CICS system that owns the transaction.

If your system is using XRF, the value returned is the generic APPLID. An application program is unaffected by a takeover from the active to the alternate.

ASRAINTRPT(*data-area***)**
Returns an 8-character data-area that contains the ILC (instruction length code) and the PIC (program interrupt code) at the point when the latest abend with a code of AICA, ASRA, ASRB, ASRD, or ASRE occurred. The field contains binary zeros if no AICA, ASRA, ASRB, ASRD, or ASRE abend occurred during the execution of the issuing transaction, or if the abend originally occurred in a remote DPL server program. When valid, the contents of the 8 bytes returned are as follows:

- ILC (2 bytes binary)
- PIC (2 bytes binary)
- filler (4 bytes binary, always zero)

ASRAKEY(*cvda***)**
Returns the execution key at the time of the last AEYD, AEYF, AICA, ASRA, or ASRB abend, if any. CVDA values are as follows:

CICSEXECKEY
This value is returned if the task was running in CICS-key at the time

of the last AEYD, AEYF, AICA, ASRA, or ASRB abend. Note that if CICS subsystem storage protection is not active, all programs run in CICS key.

USEREXECKEY
This value is returned if the task was running in user-key at the time of the last AEYD, AEYF, AICA, ASRA, or ASRB abend.

NONCICS
This value is returned if the execution key at the time of the last abend was not one of the CICS keys; for example, not key 8 or key 9.

NOTAPPLIC
This value is returned if there was not an AEYD, AEYF, AICA, ASRA, or ASRB abend.

ASRAPSW(*data-area***)**
Returns an 8-byte data area that contains the program status word (PSW) at the point when the latest abend with a code of AICA, ASRA, ASRB, ASRD, or ASRE occurred.

The field contains binary zeros if no AICA, ASRA, ASRB, ASRD, or ASRE abend occurred during the execution of the issuing transaction, or if the abend originally occurred in a remote DPL server program.

ASRAPSW16(*data-area***)**
Returns a 16-byte data area that contains the 128-bit program status word (PSW) at the point when the latest abend with a code of AICA, ASRA, ASRB, ASRD, or ASRE occurred.

The field contains binary zeros if no AICA, ASRA, ASRB, ASRD, or ASRE abend occurred during the execution of the issuing transaction, or if the abend originally occurred in a remote DPL server program.

ASRAREGS(*data-area***)**
Returns the contents of general registers 0 - 15 at the point when the latest AICA, ASRA, ASRB, ASRD, or ASRE abend occurred.

The contents of the registers are returned in the data area (64 bytes long) in the order 0, 1, ..., 14, 15.

The data area is set to binary zeros if no AICA, ASRA, ASRB, ASRD, or ASRE abend occurred during the execution of the issuing transaction, or if the abend originally occurred in a remote DPL server program.

ASRAREGS64(*data-area***)**
Returns the contents of the 64-bit general registers 0 - 15 at the point when the latest AICA, ASRA, ASRB, ASRD, or ASRE abend occurred.

The contents of the registers are returned in the data area (128 bytes long) in the order 0, 1, ..., 14, 15.

The data area is set to binary zeros if no AICA, ASRA, ASRB, ASRD, or ASRE abend occurred during the execution of the issuing transaction, or if the abend originally occurred in a remote DPL server program.

ASRASPC(*cvda***)**
Returns the type of space in control at the time of the last AEYD, AEYF, AICA, ASRA, or ASRB abend, if any. CVDA values are as follows:

SUBSPACE
This value is returned if the task was running in either its own subspace or the common subspace at the time of the last AEYD, AEYF, AICA, ASRA, or ASRB abend.

BASESPACE
This value is returned if the task was running in the base space at the time of the last AEYD, AEYF, AICA, ASRA, or ASRB abend. All tasks run in base space if transaction isolation is not active.

NOTAPPLIC
This value is returned if there was not an AEYD, AEYF, AICA, ASRA, or ASRB abend.

ASRASTG(*cvda***)**
Returns the type of storage being addressed at the time of the last AEYD, AEYF, AICA, ASRA, or ASRB abend, if any. The CVDA values are as follows:

CICS This value is returned if the storage being addressed is CICS-key storage. This can be in one of the CICS dynamic storage areas (CDSA, ECDSA, ETDSA, or GCDSA). This can be in one of the read-only dynamic storage areas (RDSA or ERDSA) when CICS is running with the NOPROTECT option on the **RENTPGM** system initialization parameter, or when storage protection is not active.

USER This value is returned if the storage being addressed is user-key storage in one of the user dynamic storage areas (UDSA, EUDSA, or GUDSA).

READONLY
This value is returned if the storage being addressed is read-only storage in one of the read-only dynamic storage areas (RDSA or ERDSA) when CICS is running with the PROTECT option on the **RENTPGM** system initialization parameter.

NOTAPPLIC
This value is returned in the following conditions:

- There is no AEYD, AEYF, AICA, ASRA, or ASRB abend found for this task.
- The affected storage in an abend is not managed by CICS.
- The ASRA abend is not caused by an 0C4 abend.

BRIDGE(*data-area***)**
Returns the 4-character TRANSID of the bridge monitor transaction that issued a **START BREXIT TRANSID** command to start the user transaction that issued this command. Blanks are returned in the following situations:

- The user transaction was not started by a bridge monitor transaction.
- This command was issued by a program started by a distributed program link (DPL) request.

Note: If the **START BREXIT** command was issued from a bridge exit, the TRANSID returned is the that of the bridge monitor that issued a **START BREXIT** naming the bridge exit.

BTRANS(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal is defined as having the background transparency capability (X'FF') or not (X'00'). If the task is not initiated from a terminal, INVREQ occurs.

CHANNEL(*data-area***)**
Returns the 16-character name of the current channel of the program, if one exists; otherwise blanks.

CMDSEC(*data-area***)**
Returns a 1-byte indicator that shows whether command security checking is defined for the current task. (X for yes, blank for no.)

COLOR(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal is defined as having the extended color capability (X'FF') or not (X'00'). If the task is not initiated from a terminal, INVREQ occurs.

CWALENG(*data-area***)**
Returns a halfword binary field that indicates the length of the common work area (CWA). If no CWA exists, a zero length is returned.

DEFSCRNHT(*data-area***)**
Returns a halfword binary variable that contains the default screen height defined for the terminal. If the task is not initiated from a terminal, INVREQ occurs.

DEFSCRNWD(*data-area***)**
Returns a halfword binary variable that contains the default screen width defined for the terminal. If the task is not initiated from a terminal, INVREQ occurs.

DELIMITER(*data-area***)**
Returns a 1-byte data-link control character for a 3600. Possible values are as follows:

X'80' Input ended with end-of-text (ETX).

X'40' Input ended with end-of-block (ETB).

X'20' Input ended with inter-record separator (IRS).

X'10' Input ended with start of header (SOH).

X'08' Transparent input.

If the task is not initiated from a terminal, INVREQ occurs.

DESTCOUNT(*data-area***)**
Returns a halfword binary field. This option has the following uses:

- Following a BMS **ROUTE** command, it shows that the value required is the number of different terminal types in the route list, and hence the number of overflow control areas that might be required.
- Within BMS overflow processing, it shows that the value required is the relative overflow control number of the destination that has encountered overflow. If this option is specified when overflow processing is not in effect, the value obtained is meaningless. If no BMS commands have been issued, INVREQ occurs.

DESTID(*data-area***)**
Returns an 8-byte identifier of the outboard destination, padded with blanks on the right to eight characters. If this option is specified before a batch data interchange command is issued in the task, INVREQ occurs.

DESTIDLENG(*data-area***)**
Returns a halfword binary length of the destination identifier obtained by DESTID. If this option is specified before a batch data interchange command is issued in the task, INVREQ occurs.

DSSCS(*data-area***)**
Returns a 1-byte indicator that shows whether the principal facility is a basic SCS data stream device (X'FF') or not (X'00').

If the task is not initiated from a terminal, INVREQ occurs.

DS3270(*data-area***)**
Returns a 1-byte indicator that shows whether the principal facility is a 3270 data stream device (X'FF') or not (X'00').

If the task is not initiated from a terminal, INVREQ occurs.

ERRORMSG(*data-area***)**
Returns the error message up to a maximum of 500 bytes that is currently referenced in the transaction abend control block for the CICS task. Following a failure of a DPL request, the message is that returned from the remote system. For messages shorter than 500 bytes the message is padded with nulls.

If no message is present, the 500 byte area contains nulls.

ERRORMSGLEN(*data-area***)**
Returns a halfword binary value representing the length of the message returned for ERRORMSG. If the message referenced in the transaction abend control block exceeds 500 bytes, the message is truncated and the length is set to 500.

If no message is present the length returned is 0.

EWASUPP(*data-area***)**
Returns a 1-byte indicator that shows whether Erase Write Alternative is supported (X'FF') or not (X'00').

If the task is not initiated from a terminal, INVREQ occurs.

EXTDS(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal accepts the 3270 extended data stream, (X'FF') or not (X'00'). Extended data stream capability is required for a terminal that supports the query feature, color, extended highlighting, programmed symbols or validation. A terminal that accepts the query structured field command also has this indicator set. If extended data stream is on, the device supports the write structured field COMMAND and Outbound Query Structured field.

For guidance information about query structured fields, see the IBM 3270 Data Stream Device Guide.

If the task is not initiated from a terminal, INVREQ occurs.

FACILITY(*data-area***)**
Returns a 4-byte identifier of the principal facility that initiated the transaction issuing this command. If this option is specified, and there is no allocated facility, INVREQ occurs.

Note: You can use the QNAME option to get the name of the transient data intrapartition queue if the transaction was initiated by expiry of a transient data trigger level.

FCI(*data-area***)**
Returns a 1-byte facility control indicator. For more information, see Codes returned by ASSIGN. This indicates the type of facility associated with the transaction; for example, X'01' indicates a terminal or logical unit. The obtained value is always returned.

GCHARS(*data-area***)**
Returns a halfword binary graphic character set global identifier (the GCSGID). The value is a number in the range 1 through 65534 representing the set of graphic characters that can be input or output at the terminal. If the task is not initiated from a terminal, INVREQ occurs.

GCODES(*data-area***)**
Returns a halfword binary code page global identifier (the CPGID). The value is a number in the range 1 through 65534 representing the EBCDIC or ASCII code page defining the code points for the characters that can be input or output at the terminal. If the task is not initiated from a terminal, INVREQ occurs.

GMMI(*data-area***)**
Returns a 1-byte indicator that shows whether a "good morning" message applies to the terminal associated with the running transaction (X'FF') or not (X'00'). If this option is specified and the current task is not associated with a terminal, the INVREQ condition occurs.

HILIGHT(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal is defined as having the extended highlight capability (X'FF') or not (X'00'). If the task is not initiated from a terminal, INVREQ occurs.

INITPARM(*data-area***)**
Returns the 60-character data-area that contains any initialization parameters that are specified for the program on the **INITPARM** system initialization parameter. The values are only returned if the name of the program that issues the command matches a program name that is specified on the **INITPARM** system initialization parameter. If there are no parameters for the program, the area is not updated and its contents are undefined. Use INITPARMLEN with **INITPARM** to determine whether a parameter was specified or not; the value that is returned by **ASSIGN INITPARM** alone cannot be used to indicate whether a system initialization parameter was specified.

INITPARMLEN(*data-area***)**
Returns a halfword binary length of the INITPARM. If there is no parameter for it, INITPARMLEN contains binary zeros.

INPARTN(*data-area***)**
Returns the 1- or 2-character name of the most recent input partition. If no map is yet positioned, or if BMS routing is in effect, or if the task is not initiated from a terminal, INVREQ occurs.

INPUTMSGLEN(*data-area***)**
Returns a halfword binary length of the terminal input string, in bytes. If there is no terminal input data, a length of zero is returned.

INVOKINGPROG(*data-area***)**
Returns the 8-character name of the application program that used the **LINK** or **XCTL** command to link or transfer control to the current program:

- If you issue the **ASSIGN INVOKINGPROG** command in a remote program that was invoked by a distributed program link (DPL) command, CICS returns the name of the program that issued the DPL command.

- If you issue the **ASSIGN INVOKINGPROG** command in an application program at the highest level, CICS returns eight blanks.
- If you issue the **ASSIGN INVOKINGPROG** command in a user-replaceable program, a Bridge Exit program or a program list table program, CICS returns eight blanks.
- If you issue the **ASSIGN INVOKINGPROG** command from a global user exit, task-related exit, or application program linked to from such an exit, CICS returns the name of the most recent invoking program that was not a global user exit or task-related user exit.

KATAKANA(*data-area***)**
Returns a 1-byte indicator that shows whether the principal facility supports Katakana (X'FF') or not (X'00'). If the task is not initiated from a terminal, INVREQ occurs.

LANGINUSE(*data-area***)**
Returns a 3-byte mnemonic code that shows the language in use. The 3-byte mnemonic has a 1:1 correspondence with the 1-byte NATLANGINUSE option. See National language codes for possible values of the code.

LDCMNEM(*data-area***)**
Returns a 2-byte logical device code (LDC) mnemonic of the destination that has encountered overflow. If this option is specified when overflow processing is not in effect, the value obtained not significant. If no BMS commands have been issued, INVREQ occurs.

LDCNUM(*data-area***)**
Returns a 1-byte LDC numeric value of the destination that has encountered overflow. This indicates the type of the LDC, such as printer or console. If this option is specified when overflow processing is not in effect, the value obtained is not significant.

LINKLEVEL(*data-area***)**
Returns a halfword binary value representing the program link level in the local system. The topmost link level is level one and for each EXEC CICS LINK the link level is incremented by one. The link level is not incremented for a language CALL statement. If a program is the target of a DPL request, the link level returned is that within the CICS region it is executing and not the wider distributed transaction. If a program is DPLed to, then link level one will be the CICS mirror program DFHMIRS.

MAJORVERSION(*data-area***)**
Returns the fullword binary value representing the major version of the current application associated with the task, which is part of the application context. If there is no application context associated with the task, then -1 is returned.

MAPCOLUMN(*data-area***)**
Returns a halfword binary number of the column on the display that contains the origin of the most recently positioned map. If no map is yet positioned, or if BMS routing is in effect, or if the task is not initiated from a terminal, INVREQ occurs.

MAPHEIGHT(*data-area***)**
Returns a halfword binary height of the most recently positioned map. If no map is yet positioned, or if BMS routing is in effect, or if the task is not initiated from a terminal, INVREQ occurs.

MAPLINE(*data-area***)**
Returns a halfword binary number of the line on the display that contains the

origin of the most recently positioned map. If no map is yet positioned, or if BMS routing is in effect, or if the task is not initiated from a terminal, INVREQ occurs.

MAPWIDTH(*data-area***)**
Returns a halfword binary width of the most recently positioned map. If no map is yet positioned, or if BMS routing is in effect, or if the task is not initiated from a terminal, INVREQ occurs.

MICROVERSION(*data-area***)**
Returns the fullword binary value representing the micro version of the current application associated with the task, which is part of the application context. If there is no application context associated with the task, then -1 is returned.

MINORVERSION(*data-area***)**
Returns the fullword binary value representing the minor version of the current application associated with the task, which is part of the application context. If there is no application context associated with the task, then -1 is returned.

MSRCONTROL(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal supports magnetic slot reader (MSR) control (`X'FF'`) or not (`X'00'`). If the task is not initiated from a terminal, INVREQ occurs.

NATLANGINUSE(*data-area***)**
Returns a 1-byte mnemonic code that shows the national language associated with the USERID for the current task (which could be the default USERID). Refer to the **SIGNON** command for an explanation of how this value is derived. (NATLANGINUSE does not show the system default language as specified on the **NATLANG** system initialization parameter.)

See National language codes for possible values of the code.

NETNAME(*data-area***)**
Returns the 8-character name of the logical unit in the z/OS Communications Server network. If the task is not initiated from a terminal, INVREQ occurs. If the principal facility is not a local terminal, CICS no longer returns a null string but the netname of the remote terminal.

If this command was issued by a user transaction that was started by a 3270 bridge transaction, the value returned is the termid of the bridge facility.

If the CICS region supports z/OS Communications Server LU aliases, the NETNAME returned by CICS could be an LU alias, either dynamically allocated by z/OS Communications Server or predefined on the **LUALIAS** parameter of a CDRSC definition.

NEXTTRANSID(*data-area***)**
Returns the 4-character next transaction identifier as set by **SET NEXTTRANSID** or **RETURN TRANSID**. It returns blanks if there are no more transactions.

NUMTAB(*data-area***)**
Returns a 1-byte number of the tabs required to position the print element in the correct passbook area of the 2980. If the task is not initiated from a terminal, INVREQ occurs.

OPCLASS(*data-area***)**
Returns, in a 24-bit string, the operator class used by BMS for routing terminal messages, as defined in the CICS segment of the External Security Manager.

OPERATION(*data-area***)**
Returns the 64 character name of the current operation associated with the task, which is part of the application context. If there is no application context associated with the task, then blanks are returned.

OPERKEYS(*data-area***)**
This option is accepted for compatibility with previous releases. If specified, a 64-bit null string is returned.

OPID(*data-area***)**
Returns the 3-character operator identification. This is used by BMS for routing terminal messages, as defined in the CICS segment of the External Security Manager.

If the task is initiated from a remote terminal, the OPID returned by this command is not necessarily that associated with the user that is signed on at the remote terminal. If you want to know the OPID of the signed on user, use the INQUIRE TERMINAL system programming command.

The OPID can also be different from the OPID of the user currently signed on, if it was changed with the SET TERMINAL command.

OPSECURITY(*data-area***)**
This option is accepted for compatibility with previous releases. If specified, a 24-bit null string is returned.

ORGABCODE(*data-area***)**
Returns a 4-byte original abend code in cases of repeated abends.

OUTLINE(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal is defined as having the field outlining capability (`X'FF'`) or not (`X'00'`). If the task is not initiated from a terminal, INVREQ occurs.

PAGENUM(*data-area***)**
Returns a halfword binary current page number for the destination that has encountered an overflow. If this option is specified when overflow processing is not in effect, the value obtained is meaningless. If no BMS commands have been issued, INVREQ occurs.

PARTNPAGE(*data-area***)**
Returns a 2-byte name of the partition that most recently caused page overflow. If no BMS commands have been issued, INVREQ occurs.

PARTNS(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal supports partitions (`X'FF'`) or not (`X'00'`). If the task is not initiated from a terminal, INVREQ occurs.

PARTNSET(*data-area***)**
Returns the name (1–6 characters) of the application partition set. A blank value is returned if there is no application partition set. If the task is not initiated from a terminal, INVREQ occurs.

PLATFORM(*data-area***)**
Returns the 64 character name of the platform associated with the task, which is part of the application context. If there is no application context associated with the task, then blanks are returned.

PRINSYSID(*data-area***)**
Returns the 4-character name by which the other system is known in the local

system; that is, the CONNECTION definition that defines the other system. For a single-session APPC device defined by a terminal definition, the returned value is the terminal identifier.

This only applies when the principal facility is one of the following:

- An MRO session to another CICS system
- An LU6.1 session to another CICS or IMS™ system
- An APPC session to another CICS system, or to another APPC system or device

If the principal facility is not an MRO, LU6.1, or APPC session, or if the task has no principal facility, INVREQ occurs.

Note: Special considerations apply generally when transaction routing. In particular an **ASSIGN PRINSYSID** command cannot be used in a routed transaction to find the name of the terminal-owning region. For more information, see CICS transaction routing.

PROCESS(*data-area***)**
Returns, if this program is running on behalf of a CICS business transaction services (BTS) activity, the 36-character name of the BTS process that contains the activity.

BTS is described in Overview of BTS.

PROCESSTYPE(*data-area***)**
Returns, if this program is running on behalf of a BTS activity, the 8-character process type of the BTS process that contains the activity.

BTS is described in Overview of BTS.

PROGRAM(*data-area***)**
Returns an 8-character name of the currently running program.

PS(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal is defined as having the programmed symbols capability (`X'FF'`) or not (`X'00'`). If the task is not initiated from a terminal, INVREQ occurs.

QNAME(*data-area***)**
Returns a 4-character name of the transient data intrapartition queue that caused this task to be initiated by reaching its trigger level. If the task is not initiated by automatic transaction initiation (ATI), INVREQ occurs.

RESSEC(*data-area***)**
Returns a 1-byte indicator that shows whether resource security checking is defined for the transaction running. (`X` for yes, blank for no.)

RESTART(*data-area***)**
Returns a 1-byte indicator that shows whether a restart of the task (`X'FF'`), or a normal start of the task (`X'00'`), has occurred.

RETURNPROG(*data-area***)**
Returns the 8-character name of the program to which control is to be returned when the current program has finished running. The values returned depend on how the current program was given control, as follows:

- If the current program was invoked by a **LINK** command, including a distributed program link, RETURNPROG returns the same name as INVOKINGPROG.

- If the current program was invoked by an **XCTL** command, RETURNPROG returns the name of the application program in the chain that last issued a LINK command.

  If the program that invoked the current program with an **XCTL** command is at the highest level, CICS returns eight blanks.
- If the **ASSIGN RETURNPROG** command is issued in the program at the top level, CICS returns eight blanks.
- If the **ASSIGN RETURNPROG** command is issued in a user-replaceable module, or a program list table program, CICS returns eight blanks.
- If the **ASSIGN RETURNPROG** command is issued in a global user exit, task-related exit, or application program linked to from such an exit, CICS returns the name of the program that control is returned to when all intermediate global user exit and task-related user exit programs have completed.

SCRNHT(*data-area***)**
Returns a halfword binary variable that contains the height of the 3270 screen defined for the current task. If the task is not initiated from a terminal, INVREQ occurs.

SCRNWD(*data-area***)**
Returns a halfword binary variable that contains the width of the 3270 screen defined for the current task. If the task is not initiated from a terminal, INVREQ occurs.

SIGDATA(*data-area***)**
Returns a 4-byte character string that contains the inbound signal data received from a logical unit. If the task is not initiated from a terminal, INVREQ occurs.

SOSI(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal is defined as having the mixed EBCDIC/DBCS fields capability (`X'FF'`) or not (`X'00'`). The DBCS subfields within an EBCDIC field are delimited by SO (shift-out) and SI (shift-in) characters. If the task is not initiated from a terminal, INVREQ occurs.

STARTCODE(*data-area***)**
Returns a 2-character value that indicates how the transaction that issued the request was started. Possible values are as follows:

Code	Transaction started by
D	A distributed program link (DPL) request that did not specify the `SYNCONRETURN` option. The task cannot issue I/O requests against its principal facility, or any sync point requests.
DS	A distributed program link (DPL) request, as in code D, that did specify the `SYNCONRETURN` option. The task can issue sync point requests.
QD	Transient data trigger level.
S	START command that did not pass data in the `FROM` option. It might or might not have passed a channel.
SD	START command that passed data in the `FROM` option.
SZ	**FEPI START** command.
TD	Terminal input or permanent transid.
U	User-attached task.

STATIONID(*data-area***)**
Returns a 1-byte station identifier of a 2980. If the task is not initiated from a terminal, INVREQ occurs.

SYSID(*data-area***)**
Returns the 4-character name given to the local CICS system. This value can be specified in the SYSID option of a file control, interval control, temporary storage, or transient data command, in which case the resource to be accessed is assumed to be on the local system.

TASKPRIORITY(*data-area***)**
Returns a halfword binary filed that indicates the current priority of the issuing task (0–255). When the task is first attached, this is the sum of the user, terminal, and transaction priorities. This value can be changed during execution by a **CHANGE TASK** command.

TCTUALENG(*data-area***)**
Returns a halfword binary length of the terminal control table user area (TCTUA). If no TCTUA exists, a zero length is returned.

TELLERID(*data-area***)**
Returns a 1-byte teller identifier of a 2980. If the task is not initiated from a terminal, INVREQ occurs.

TERMCODE(*data-area***)**
Returns a 2-byte code giving the type and model number of the terminal associated with the task.

The first byte is a code identifying the terminal type, derived from the TERMINAL resource. For a description of the resource attributes, see TERMINAL attributes . The second byte is a single-character model number as specified in the TERMMODEL attribute.

The meanings of the type codes are given in Codes returned by ASSIGN.

TERMPRIORITY(*data-area***)**
Returns a halfword binary terminal priority (0–255).

TEXTKYBD(*data-area***)**
Returns a 1-byte indicator that shows whether the principal facility supports TEXTKYBD (X'FF') or not (X'00'). If the task is not initiated from a terminal, INVREQ occurs.

TEXTPRINT(*data-area***)**
Returns a 1-byte indicator that shows whether the principal facility supports TEXTPRINT (X'FF') or not (X'00'). If the task is not initiated from a terminal, INVREQ occurs.

TRANPRIORITY(*data-area***)**
Returns a halfword binary transaction priority (0–255).

TWALENG(*data-area***)**
Returns a halfword binary length of the transaction work area (TWA). If no TWA exists, a zero length is returned.

UNATTEND(*data-area***)**
Returns a 1-byte indicator that shows whether the mode of operation of the terminal is unattended, that is to say that no person is attending the terminal. These indicators are X'FF' for unattended and X'00' for attended. If the task is not initiated from a terminal, INVREQ occurs.

USERID(*data-area***)**
Returns an 8-byte user ID of the signed-on user. If no user is explicitly signed

on, CICS returns the default user ID. Special considerations apply if you are using an intercommunication environment. See the Getting started with intercommunication for more information about the **ASSIGN** command for LUTYPE6.1, APPC, and MRO.

USERNAME(*data-area***)**
Returns a 20-character name of the user obtained from the external security manager (ESM).

USERPRIORITY(*data-area***)**
Returns a halfword binary operator priority (0–255).

VALIDATION(*data-area***)**
Returns a 1-byte indicator that shows whether the terminal is defined as having the validation capability (`X'FF'`) or not (`X'00'`). Validation capability consists of the mandatory fill, mandatory enter, and trigger attributes. If the task is not initiated from a terminal, INVREQ occurs.

Conditions

The ASSIGN command always returns the exception condition INVREQ under CECI or in a REXX program. Even though CECI or the REXX program might return the information you requested correctly, it also attempts to get information from other options, some of which are invalid.

16 INVREQ
RESP2 values:

1 The task does not have a signed-on user.

2 No BMS command has yet been issued, BMS routing is in effect, or no map has yet been positioned.

3 No batch data interchange (BDI) command has yet been issued.

4 The task is not initiated by automatic transaction initiation (ATI).

5 The task is not associated with a terminal; or the task has no principal facility; or the principal facility is not an MRO, LU6.1, or APPC session.

6 A CICS BTS request was issued from outside the CICS BTS environment. Therefore, the transaction is not running on behalf of a BTS activity.

200 Command syntax options are not allowed in a server program invoked by a distributed program link.

Default action: terminate the task abnormally.

Examples

An example of RETURNPROG:

```
Program A links to program B
Program B links to program C
Program C transfers control to program D
Program D issues an ASSIGN RETURNPROG command,
  and CICS returns the name of Program B.
```

Codes returned by ASSIGN

The **ASSIGN** command can return codes in the TERMCODE and FCI options. The format and meaning of the codes are described.

ASSIGN TERMCODE

The following list shows you the meanings of the terminal type codes in the first byte of the data area returned by the TERMCODE option of the ASSIGN command.

The codes are derived from the DEVICE attribute of the TYPETERM resource. The second byte of the data area contains a model number in character form, as set by the TERMMODEL attribute of the TYPETERM resource definition.

The codes are listed here as both bit patterns and hexadecimal values.

Code		Meaning
....1	X'01'	7770
.... ..1.	X'02'	System 7
.... 1...	X'08'	Console
...1 ..1.	X'12'	Sequential disk
...1 .1..	X'14'	Magnetic tape
...1 1...	X'18'	Card reader or line printer
...1 1..1	X'19'	Spooling system printer
...1 1.1.	X'1A'	Spooling internal reader
..1.	X'20'	Hard-copy terminals
..1. ...1	X'21'	Model 33/35 TWX
..1. ..1.	X'22'	Teletypewriter
..1. .1..	X'24'	1050
..1. 1...	X'28'	2740
..1. 1.1.	X'2A'	2741 Correspondence
..1. 1.11	X'2B'	2741 EBCDIC
.1..	X'40'	Video terminals
.1.. ...1	X'41'	2260 local
.1.. 1...	X'48'	2260 remote
.1.. 1.1.	X'4A'	1053
.1.. 11..	X'4C'	2265
1...	X'80'	Bisynchronous
1... ..1.	X'82'	2770
1... .1..	X'84'	2780
1... .1.1	X'85'	3780
1... .11.	X'86'	2980
1... 1...	X'88'	3735
1... 1..1	X'89'	3740
1... 1.1.	X'8A'	3600 bisynchronous
1..1 ...1	X'91'	3277 remote
1..1 ..1.	X'92'	3275 remote
1..1 1..1	X'99'	3277 local
1.1.	X'A0'	Bisynchronous - programmable
1.1. ...1	X'A1'	System/3
1.1. .1..	X'A4'	z Systems™
1.1. .11.	X'A6'	System/7 with BSCA
1.11	X'B0'	SDLC device class
1.11 ...1	X'B1'	3601
1.11 ..1.	X'B2'	3614

Code		Meaning
1.11 .1..	X'B4'	3790
1.11 .1.1	X'B5'	3790 User program
1.11 .11.	X'B6'	3790 SCS printer
1.11 1...	X'B8'	3650 Pipeline
1.11 1..1	X'B9'	3653 Host conversational
1.11 1.1.	X'BA'	3650 Attached 3270 HC
1.11 1.11	X'BB'	3650 User program
1.11 11.1	X'BD'	Contention logical unit
1.11 111.	X'BE'	Interactive logical unit
1.11 1111	X'BF'	Batch logical unit
11..	X'C0'	LUTYPE 6 **Note:** An ASSIGN TERMCODE for an ISC session returns X'C0' for LUTYPE 6. An INQUIRE CONNECTION then determines whether this ISC connection is using LUTYPE6.1 or APPC protocols.
11.. ...1	X'C1'	LUTYPE 4
11.1 ...1	X'D1'	ISC MM conversation
11.1 ..1.	X'D2'	LUC mode group entry
11.1 ..11	X'D3'	LUC session

ASSIGN FCI

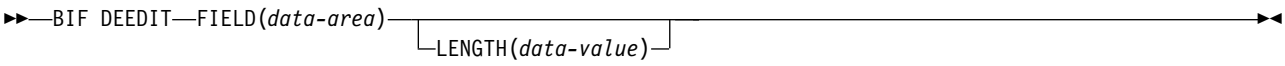
The following list shows you the meanings of the facility control indicator codes in the data area returned by the FCI option of the ASSIGN command.

Code		Meaning
....	X'00'	None
....1	X'01'	Terminal facility indicator
.... ..1.	X'02'	KCP macro file mask
.... .1..	X'04'	Interval control indicator
.... 1...	X'08'	Destination control indicator
...1	X'10'	AID facility mask
111.	X'E0'	reserved

BIF DEEDIT

Provide the DEEDIT function that removes special characters from an EBCDIC data field.

BIF DEEDIT



Condition: LENGERR

This command is threadsafe.

Description

BIF DEEDIT provides the built-in function DEEDIT. It specifies that alphabetic and special characters are removed from an EBCDIC data field, and the remaining digits right-aligned and padded to the left with zeros as necessary.

If the field ends with a minus sign or a carriage-return (CR), a negative zone (X'D') is placed in the rightmost (low-order) byte.

If the zone portion of the rightmost byte contains one of the characters X'A' through X'F', and the numeric portion contains one of the hexadecimal digits X'0' through X'9', the rightmost byte is returned unaltered (see the example). This permits the application program to operate on a zoned numeric field. The returned value is in the field that initially contained the unedited data.

A 1 byte field is returned unaltered, no matter what the field contains.

Options

FIELD(*data-area***)**
specifies the field to be edited.

LENGTH(*data-value***)**
specifies the field length in bytes.

Conditions

22 LENGERR
Occurs if the LENGTH value is less than 1.

Default action: terminate the task abnormally.

Examples

```
EXEC CICS BIF DEEDIT
     FIELD(CONTG)
     LENGTH(9)
```

This removes all characters other than digits from CONTG, a 9 byte field, and returns the edited result in that field to the application program.

Two examples of the contents of CONTG before and after execution of the command are:

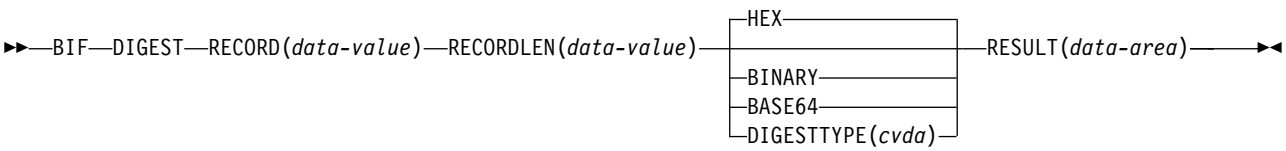
```
Original value      Returned value
14-6704/B             00146704B
$25.68                000002568
```

A decimal point is an EBCDIC special character and as such is removed.

BIF DIGEST

Calculate the SHA-1 digest of a string of data.

BIF DIGEST



Conditions: INVREQ, LENGERR

This command is threadsafe.

Description

The **BIF DIGEST** command is a CICS built-in function that calculates the SHA-1 digest of a string of data. The result can be returned as binary (20 bytes long), hexadecimal (40 bytes long), or base64-encoded (28 bytes long). The SHA-1 digest is a cryptographically strong checksum of the string, so for practical purposes it is unique for each string.

This command uses z/Architecture® message security assist (MSA) functions that require z Systems cryptographic hardware with CP Assist for Cryptographic Functions (CPACF). For more information, see the *z/OS Cryptographic Services Integrated Cryptographic Service Facility Overview*.

Options

RECORD(*data-value***)**
Specifies the string of data for which the digest is to be calculated.

RECORDLEN(*data-value***)**
Specifies the length, as a fullword binary value, of the data string.

DIGESTTYPE(*cvda***)**
Specifies the format in which the digest is returned.

HEX Hexadecimal, which produces a result 40 bytes long, encoded as hexadecimal characters (0 - 9, A - F).

BINARY
Binary, which produces a result 20 bytes long.

BASE64
Base64 encoding, which produces a result 28 bytes long, using the characters A - Z, a - z, 0 - 9, +, /, =.

RESULT(*data-area***)**
Returns the SHA-1 digest of the data string in the format specified by the DIGESTTYPE option. The length of the result depends on the requested format.

Conditions

16 INVREQ
RESP2 values:

1 DIGESTTYPE has an invalid CVDA value.

3 z/Architecture message security assist (MSA) is not available.

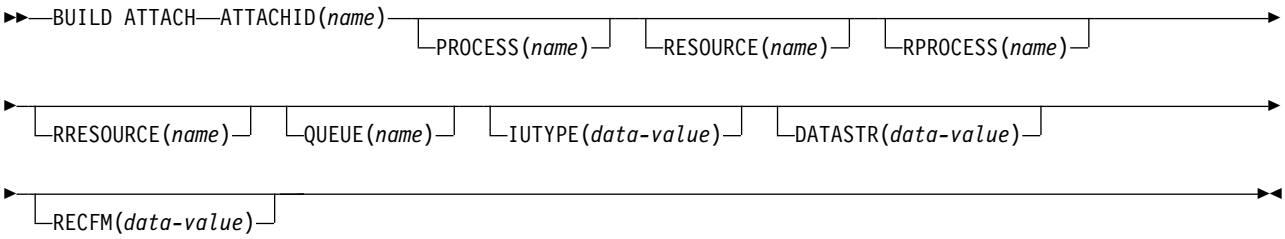
22 LENGERR
RESP2 values:

2 The RECORDLEN value is less than 1.

BUILD ATTACH (LUTYPE6.1)

Specify values for an LUTYPE6.1 attach header.

BUILD ATTACH (LUTYPE6.1)



Description

BUILD ATTACH (LUTYPE6.1) specifies a set of values to be placed in the named attach header control block. This control block contains values that are to be sent in an LUTYPE6.1 attach FMH (Function Management Header) that is constructed by CICS, and is sent only when a SEND ATTACHID or CONVERSE ATTACHID command is executed. The specified values override existing values in the control block; unspecified values are set to default values.

Options

ATTACHID(*name***)**
specifies that the set of values is to be placed in an attach header control block identified by the specified name (1–8 characters).

DATASTR(*data-value***)**
corresponds to the data stream profile field, ATTDSP, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no particular significance is attached by CICS to the data stream profile field in an attach FMH. For most CICS applications, the option can be omitted.

. For details of communication between a CICS system and any other subsystem, including details of structured fields and logical record

management, refer to documentation supplied by the subsystem about how to use the data stream profile field in an attach FMH.

The "data-value" is a halfword binary. Only the low-order byte is used. The SNA-defined meanings of the bits are as follows:

```
0–7    reserved - must be set to zero
8-11   0000 - user-defined
       1111 - SCS data stream
       1110 - 3270 data stream
       1101 - structured field
       1100 - logical record management
12-15  defined by the user if bits 8-11
       are set to 0000; otherwise reserved
       (must be set to zero)
```

A value of "structured field" indicates that chains begin with four bytes of data that are used to interpret the following data: overall length (2 bytes), class identifier (1 byte), and subclass identifier (1 byte). A value of "logical record management" indicates that chains can be split into separate fields by the data receiver.

If the option is omitted, a value of "user-defined" is assumed.

IUTYPE(*data-value***)**
corresponds to the interchange unit field, ATTIU, in an LUTYPE6.1 attach FMH.

The "data-value" is a halfword binary. Only the low-order 7 bits are used. The SNA-defined meanings of the bits are as follows:

```
0–10   reserved - must be set to zero
11     0 - not end of multichain
           interchange unit
       1 - end of multichain
           interchange unit
12,13  reserved - must be set to zero
14,15  00 - multichain interchange
            unit
       01 - single-chain
            interchange unit
       10 - reserved
       11 - reserved
```

If the option is omitted, values of "not end of multichain interchange unit" and "multichain interchange unit" are assumed.

PROCESS(*name***)**
corresponds to the process name, ATTDPN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, a transaction running in one system can acquire a session to the second system and can identify the transaction to be attached; in the second system, the identification is carried in the first chain of data sent across the session.

In general, the first four bytes of data identify the transaction to be attached. However an attach FMH, identifying the transaction to be attached, can be built and sent; the PROCESS option is used to specify the transaction name. (Note that the receiving CICS system uses just the first four bytes of the process name as a transaction name.)

No significance is attached by CICS to process names in attach FMHs sent in chains of data other than the first.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the process name field in an attach FMH.

QUEUE(*name***)**
corresponds to the queue name, ATTDQN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no significance is attached by CICS to the queue name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the queue name field in an attach FMH.

RECFM(*data-value***)**
corresponds to the deblocking algorithm field, ATTDBA, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no particular significance is attached by CICS to the deblocking algorithm field in an attach FMH. For most CICS applications, the option can be omitted.

The "data-value" is a halfword binary value. Only the low-order byte is used. The SNA-defined meanings of the bits are as follows:

```
0–7   reserved - must be set to zero
8-15  X'00' - reserved
      X'01' - variable-length
              variable-blocked
      X'02' - reserved
      X'03' - reserved
      X'04' - chain of RUs
      X'05'through X'FF' - reserved
```

If the option is omitted, a value of "chain of RUs" is assumed.

RESOURCE(*name***)**
corresponds to the resource name, ATTPRN, in an LUTYPE6.1 attach FMH.

RPROCESS(*name***)**
corresponds to the return-process name, ATTRDPN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no significance is attached by CICS to the return-process name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the return-process name field in an attach FMH.

RRESOURCE(*name***)**
corresponds to the return-resource name, ATTRPRN, in an LUTYPE6.1 attach FMH.

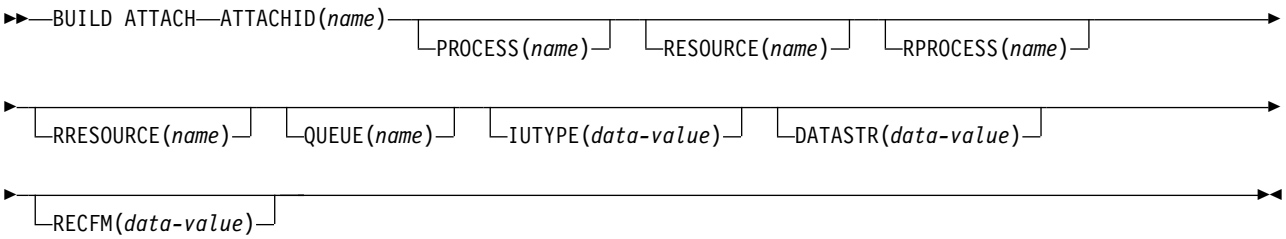
For communication between two CICS systems, no significance is attached by CICS to the return-resource name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the return-resource name field in an attach FMH.

BUILD ATTACH (MRO)

Specify values for an MRO attach header.

BUILD ATTACH (MRO)



Description

BUILD ATTACH (MRO) specifies a set of values to be placed in the named attach header control block. This control block contains values that are to be sent in an MRO attach FMH (Function Management Header) that is constructed by CICS, and is sent only when a SEND ATTACHID or CONVERSE ATTACHID command is executed. The specified values override existing values in the control block; unspecified values are set to default values.

For more information about MRO and IRC, see Introduction to CICS intercommunication.

Options

ATTACHID(*name***)**
specifies that the set of values is to be placed in an attach header control block identified by the specified name (1–8 characters).

DATASTR(*data-value***)**
corresponds to the data stream profile field, ATTDSP, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no particular significance is attached by CICS to the data stream profile field in an attach FMH. For most CICS applications, the option can be omitted.

The “data-value” is a binary halfword. Only the low-order byte is used. The SNA-defined meanings of the bits are as follows:

```
0–7    reserved - must be set to zero
8-11   0000 - user-defined
       1111 - SCS data stream
       1110 - 3270 data stream
       1101 - structured field
       1100 - logical record management
12-15  defined by the user if bits 8-11
       are set to 0000; otherwise reserved
       (must be set to zero)
```

A value of “structured field” indicates that chains begin with four bytes of data that are used to interpret the following data; overall length (2 bytes), class

identifier (1 byte), and subclass identifier (1 byte). A value of "logical record management" indicates that chains can be split into separate fields by the data receiver.

If the option is omitted, a value of "user-defined" is assumed.

IUTYPE(*data-value***)**
corresponds to the interchange unit field, ATTIU, in an LUTYPE6.1 attach FMH.

The "data-value" is a halfword binary. Only the low-order 7 bits are used. The SNA-defined meanings of the bits are as follows:

```
0–10   reserved - must be set to zero
11     0 - not end of multichain interchange unit
       1 - end of multichain interchange unit
12,13  reserved - must be set to zero
14,15  00 - multichain interchange unit
       01 - single chain interchange unit
       10 - reserved
       11 - reserved
```

If the option is omitted, values of "not end of multichain interchange unit" and "multichain interchange unit" are assumed.

PROCESS(*name***)**
corresponds to the process name, ATTDPN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, a transaction running in one system can acquire a session to the second system and can identify the transaction to be attached; in the second system the identification is carried in the first chain of data sent across the session. In general, the first four bytes of data identify the transaction to be attached. However an attach FMH, identifying the transaction to be attached, can be built and sent; the PROCESS option is used to specify the transaction name. (Note that the receiving CICS system uses just the first four bytes of the process name as a transaction name.)

No significance is attached by CICS to process names in attach FMHs sent in chains of data other than the first.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the process name field in an attach FMH.

QUEUE(*name***)**
corresponds to the queue name, ATTDQN, in an attach FMH.

For communication between two CICS systems, no significance is attached by CICS to the queue name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the queue name field in an attach FMH.

RECFM(*data-value***)**
corresponds to the deblocking algorithm field, ATTDBA, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no particular significance is attached by CICS to the deblocking algorithm field in an attach FMH. For most CICS applications, the option can be omitted.

The “data-value” is a halfword binary value. Only the low-order 8 bits are used. The SNA-defined meanings of the bits are as follows:

```
0–7   reserved - must be set to zero
8-15  X'00' - reserved
      X'01' - variable-length
              variable-blocked
      X'02' - reserved
      X'03' - reserved
      X'04' - chain of RUs
      X'05' to X'FF' - reserved
```

If the option is omitted, a value of “chain of RUs” is assumed.

RESOURCE(*name*)
corresponds to the resource-name, ATTPRN, in an LUTYPE6.1 attach FMH.

RPROCESS(*name*)
corresponds to the return-process name, ATTRDPN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no significance is attached by CICS to the return-process name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the return-process name field in an attach FMH.

RRESOURCE(*name*)
corresponds to the return-resource name, ATTRPRN, in an LUTYPE6.1 attach FMH.

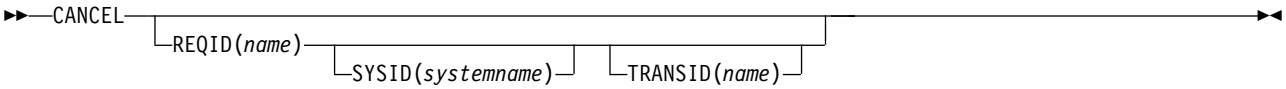
For communication between two CICS systems, no significance is attached by CICS to the return-resource name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the return-resource name field in an attach FMH.

CANCEL

Cancel interval control requests.

CANCEL



Conditions: ISCINVREQ, NOTAUTH, NOTFND, SYSIDERR

Note for dynamic transaction routing: Using CANCEL with REQID (of a POST, DELAY, or START) could create intertransaction affinities that adversely affect the use of dynamic transaction routing. See Affinity for more information about transaction affinities.

Description

CANCEL cancels a previously issued DELAY, POST, or START command. The CANCEL command cannot be used to remove a request that is queued locally. If you include the SYSID option, the command is shipped to a remote system. If you omit SYSID, the TRANSID option, if present, indicates where the command is to be executed. The effect of the cancelation varies depending on the type of command being canceled, as follows:

- A DELAY command can be canceled only before it has expired, and only by a task other than the task that issued the DELAY command (which is suspended for the duration of the request). The REQID used by the suspended task must be specified. The effect of the cancelation is the same as an early expiration of the original DELAY. That is, the suspended task becomes dispatchable as though the original expiration time had been reached.
- When a POST command issued by the same task is to be canceled, no REQID need be specified. Cancelation can be requested either before or after the original request has expired. The effect of the cancelation is as if the original request had never been made.
- When a POST command issued by another task is to be canceled, the REQID of that command must be specified. The effect of the cancelation is the same as an early expiration of the original POST request. That is, the timer event control area for the other task is posted as though the original expiration time had been reached.
- When a START command is to be canceled, the REQID associated with the original command must be specified. The effect of the cancelation is as if the original command had never been issued. The cancelation is effective only before the original command has been honored.
- When you use a START command with the PROTECT option, CANCEL will cancel the START command only if the START command has been committed.

Note: A NOTFND response to a CANCEL command of a START with REQID signifies that the start request is no longer outstanding. It does not imply that the task to be started has completed by this point in time; neither does it imply that the started task has issued a RETRIEVE command to read the FROM data from the REQID queue. A subsequent START command reusing the same REQID value may fail with an AEIQ abend (IOERR condition), if the REQID queue still exists at this time.

Options

REQID(*name***)**
specifies a name (1–8 characters), which should be unique, to identify a command. This name is used as a temporary storage identifier. The temporary storage queue thus identified must be defined as a local queue on the CICS system where the CANCEL command is processed.

This option cannot be used to cancel a POST command issued by the same task (for which, the REQID option is ignored if it is specified).

SYSID(*systemname***)**
(remote systems only) specifies the name (1–4 characters) of the system for the CANCEL command.

TRANSID(*name***)**
specifies the symbolic identifier (1–4 characters) of a transaction to be used to

determine where the CANCEL command is to be executed, if SYSID is not specified. If the TRANSID is defined as REMOTE, the CANCEL request is function-shipped to the remote system.

Conditions

54 ISCINVREQ
occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

70 NOTAUTH
occurs when a resource security check has failed on the specified TRANSID or on the TRANSID of the START command that corresponds to the request identification.

Default action: terminate the task abnormally.

13 NOTFND
occurs if the request identifier specified fails to match an unexpired interval control command.

Default action: terminate the task abnormally.

53 SYSIDERR
occurs when the SYSID option specifies a name that is neither the local system nor a remote system (made known to CICS by defining a CONNECTION). It also occurs when the link to the remote system is closed.

Default action: terminate the task abnormally.

CANCEL (BTS)

Cancel a BTS activity or process.

CANCEL (BTS)



Conditions: ACTIVITYBUSY, ACTIVITYERR, INVREQ, IOERR, LOCKED, NOTAUTH, PROCESSBUSY, PROCESSERR

Description

CANCEL (BTS) forces a BTS activity or process, and all its descendant activities, into COMPLETE mode.

Options

ACQACTIVITY
specifies that the activity to be canceled is the one that the current unit of work has acquired by means of an ACQUIRE ACTIVITYID command.

ACQPROCESS
specifies that the process that the current unit of work has acquired is to be canceled.

ACTIVITY(data-value)
specifies the name (1–16 characters) of the child activity to be canceled.

Conditions

107 ACTIVITYBUSY
RESP2 values:

19 One or more of the descendant activities of the activity to be canceled are inaccessible or in CANCELLING mode.

109 ACTIVITYERR
RESP2 values:

8 The activity named on the ACTIVITY option could not be found.

14 The activity to be canceled is not in INITIAL or DORMANT mode.

16 INVREQ
RESP2 values:

4 The ACTIVITY option was used to name a child activity, but the command was issued outside the scope of a currently-active activity.

15 The ACQPROCESS option was used, but the issuing task has not acquired a process.

24 The ACQACTIVITY option was used, but the issuing task has not acquired an activity.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access the file associated with the BTS repository data set on which details of the process or activity are stored.

106 PROCESSBUSY
RESP2 values:

13 One or more of the activities that make up the process to be canceled are inaccessible or in CANCELLING mode.

108 PROCESSERR
RESP2 values:

9 The process—type could not be found.

14 The process to be canceled is not in INITIAL, DORMANT, or COMPLETE mode.

Activities

The only activities a program can cancel are as follows:

- If it is running as the activation of an activity, its own child activities. It can cancel several of its child activities within the same unit of work.
- The activity it has acquired, by means of an ACQUIRE ACTIVITYID command, in the current unit of work.

To be canceled successfully, an activity must be in INITIAL or DORMANT mode. CICS tries to cancel activities synchronously. However, if one or more descendant activities of the activity to be canceled are inaccessible (due, for example, to the failure of a communications link):

- The subtree of descendant activities is canceled asynchronously.
- The activity to be canceled is placed in CANCELLING mode.

The completion event associated with a canceled activity is not deleted from the parent's event pool. On normal completion of this command, the activity still exists, and can be reset and run again, if necessary.

When an acquired activity is canceled, its parent is reactivated because of the firing of the canceled activity's completion event.

Processes

The only process a program can cancel is the one it has acquired in the current unit of work. If it does so, it cannot acquire another process within the current unit of work.

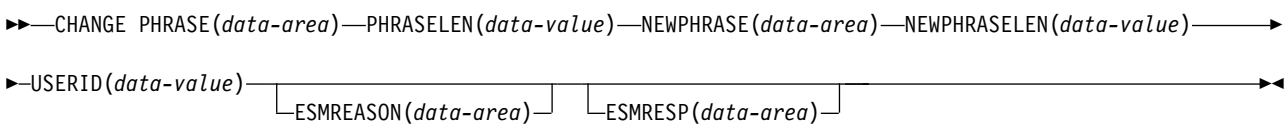
To be canceled successfully, a process must be in INITIAL, DORMANT, or COMPLETE mode.

CICS tries to cancel the process synchronously, in the way described for activities.

CHANGE PHRASE

Change the password or password phrase recorded by an external security manager (ESM) for a specified user ID.

CHANGE PHRASE



Conditions: INVREQ, LENGERR, NOTAUTH, USERIDERR

This command is threadsafe.

Description

A user ID can have both a password and a password phrase. If PHRASELEN is between 1 and 8 characters, the phrase is treated as a password. If the length is between 9 and 100 characters, it is treated as a password phrase. You cannot use a

1- to 8-character password to change a password phrase. Similarly, you cannot use a 9- to 100-character password phrase to change a 1- to 8-character password.

Unlike the SIGNON command, CHANGE PHRASE does not depend upon the principal facility, therefore it can be issued in non-terminal environments such as Web applications and APPC sessions.

Attention: To ensure that passwords are not revealed in system or transaction dumps, clear the password or password phrase fields on the EXEC CICS commands that have a password or password phrase option as soon as possible after use.

Options

Options ESMRESP and ESMREASON return the response and reason codes, if any, from the external security manager.

ESMREASON(*data-area***)**
returns the reason code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF®, this field is the RACF reason code.

ESMRESP(*data-area***)**
returns the response code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF return code.

NEWPHRASE(*data-area***)**
specifies an optional 1- to 8-character new password or a 9- to 100-character new password phrase required by the ESM. The password is changed only if the current password is correctly specified. The password phrase is changed only if the current password phrase is correctly specified.

NEWPHRASELEN(*data-area***)**
specifies the length, as a fullword binary value, of the new password or password phrase.

PHRASE(*data-area***)**
specifies the current password or password phrase of the specified user ID.

PHRASELEN(*data-area***)**
specifies the length, as a fullword binary value, of the current password or password phrase.

USERID(*data-value***)**
specifies the user ID of the user whose password or password phrase is being changed.

Conditions

16 INVREQ
RESP2 values:

2 You cannot use a password to change a password phrase or a password phrase to change a password.

13 The external security manager has issued an unknown return code in ESMRESP.

18 The CICS external security manager interface is not initialized.

29 The external security manager is not responding.

Default action: terminate the task abnormally.

22 LENGERR
RESP2 values:

1 PHRASELEN was out-of-range.

2 NEWPHRASELEN was out-of-range.

70 NOTAUTH
RESP2 values:

2 The supplied password or password phrase is wrong. If the external security manager is RACF , the revoke count maintained by RACF is incremented.

4 The new password or password phrase is not acceptable.

19 The user ID is revoked.

20 The connection to the user's default group has been revoked.

22 The change password request failed during SECLABEL processing.

31 The user is revoked in the connection to the default group.

Default action: terminate the task abnormally.

69 USERIDERR
RESP2 values:

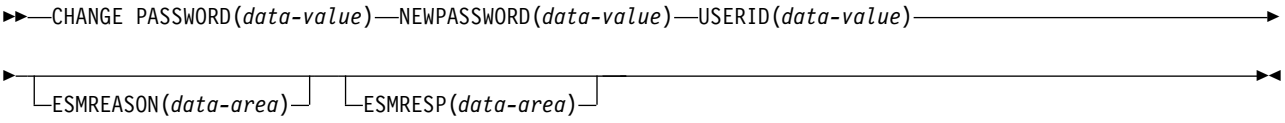
8 The USERID is not known to the external security manager.

Default action: terminate the task abnormally.

CHANGE PASSWORD

Change the password recorded by an external security manager (ESM) for a specified user ID.

CHANGE PASSWORD



Conditions: INVREQ, NOTAUTH, USERIDERR

This command is threadsafe.

Description

Unlike the SIGNON command, CHANGE PASSWORD does not depend upon the principal facility, therefore it can be issued in non-terminal environments such as Web applications and APPC sessions.

Attention: You should clear the password fields on the EXEC CICS commands that have a password option as soon as possible after use. This is to ensure that passwords are not revealed in system or transaction dumps.

Options

Options ESMRESP and ESMREASON return the response and reason codes, if any, from the external security manager.

ESMREASON(*data-area***)**
returns the reason code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF reason code.

ESMRESP(*data-area***)**
returns the response code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF return code.

NEWPASSWORD(*data-value***)**
specifies the new password, 8 characters, for the specified userid. The password is changed only if the current password is correctly specified.

PASSWORD(*data-value***)**
specifies the current password, 8 characters, for the specified userid.

USERID(*data-value***)**
specifies the userid, 8 characters, of the user whose password is being changed.

Conditions

16 INVREQ
RESP2 values:

13 There is an unknown return code in ESMRESP from the external security manager.

18 The CICS external security manager interface is not initialized.

29 The external security manager is not responding.

Default action: terminate the task abnormally.

70 NOTAUTH
RESP2 values:

2 The supplied password is wrong. If the external security manager is RACF , the revoke count maintained by RACF is incremented.

4 The new password is not acceptable.

19 The USERID is revoked.

22 The change password request failed during SECLABEL processing.

31 The user is revoked in the connection to the default group.

Default action: terminate the task abnormally.

69 USERIDERR
RESP2 values:

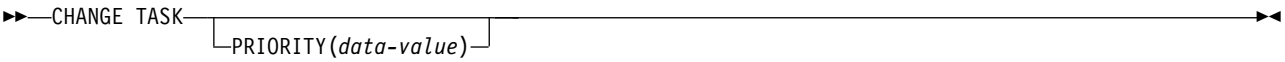
8 The USERID is not known to the external security manager.

Default action: terminate the task abnormally.

CHANGE TASK

Change priority of a task.

CHANGE TASK



Condition: INVREQ

This command is threadsafe.

Description

CHANGE TASK changes the priority of the issuing task. It has immediate effect (unlike SET TASK), because control is relinquished during execution of the command so that the current task has to be redispatched. The redispatch does not happen until tasks that are of higher or equal priority, and that are also ready to run, are dispatched.

If you omit the PRIORITY option, the task does not lose control and the priority remains the same. This is effectively a no-op.

Options

PRIORITY(*data-value***)**
specifies a fullword binary value in the range 0–255, defining the priority of the task. You can also have a value of -1 but this does not change the priority or cause a redispatch.

Conditions

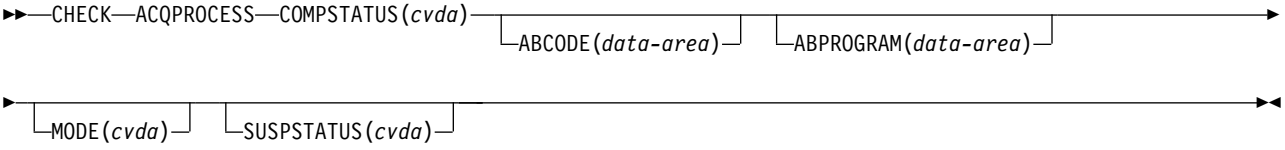
16 INVREQ
RESP2 values:

1 Your PRIORITY value is outside the range -1–255.

CHECK ACQPROCESS

Check the completion status of a BTS process.

CHECK ACQPROCESS



Conditions: INVREQ

Description

CHECK ACQPROCESS returns the completion status of the currently-acquired BTS process. Typically, it is used to check the success of a previous RUN ACQPROCESS or LINK ACQPROCESS command. It allows the requestor to discover whether the process completed successfully, or whether, for example, it needs to be reactivated in order to complete its processing.

The only process a program can check is the one that it has acquired in the current unit of work - see Acquiring processes and activities .

The RESP and RESP2 options on this command reflect whether the command is understood by CICS - for example, PROCESSERR occurs if the process is not currently acquired by the requestor.

The COMPSTATUS option returns a CVDA value indicating the completion status of the process's root activity - for example, NORMAL is returned if the root activity has successfully completed all its processing steps, while INCOMPLETE is returned if it has returned from an activation but needs to be reattached in order to complete its processing.

Options

ABCODE(data-area)
returns, if the process's root activity terminated abnormally, the 4-character abend code.

ABPROGRAM(data-area)
returns, if the process's root activity terminated abnormally, the 8-character name of the program that was in control at the time of the abend.

ACQPROCESS
specifies that the process that is currently acquired by the requestor is to be checked.

COMPSTATUS(cvda)
indicates the completion status of the process. CVDA values are:

ABEND
The program that implements the process's root activity abended. Any children of the root activity have been canceled.

FORCED
The process was forced to complete—for example, it was canceled with a CANCEL ACQPROCESS command.

INCOMPLETE
The process is incomplete. This could mean:

- That it has not yet been run
- That it has returned from one or more activations but needs to be reattached in order to complete all its processing steps
- That it is currently active.

NORMAL
The process completed successfully.

MODE(cvda)
indicates the processing state of the process. CVDA values are:

ACTIVE
An activation of the process is running.

CANCELLING
CICS is waiting to cancel the process. A CANCEL ACQPROCESS command has been issued, but CICS cannot cancel the process immediately because one or more of the root activity's children are inaccessible.

COMPLETE
The process has completed.

DORMANT
The process is waiting for an event to fire its next activation.

INITIAL
No RUN or LINK command has yet been issued against the process.

SUSPSTATUS(cvda)
indicates whether the process is currently suspended. CVDA values are:

SUSPENDED
The process is currently suspended. If a reattachment event occurs, it will not be reactivated.

NOTSUSPENDED
The process is not currently suspended. If a reattachment event occurs, it will be reactivated.

Conditions

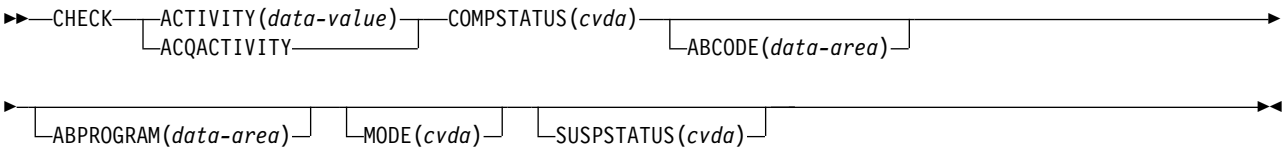
16 INVREQ
RESP2 values:

15 The unit of work that issued the request has not acquired a process.

CHECK ACTIVITY

Check the completion status of a BTS activity.

CHECK ACTIVITY



Conditions: ACTIVITYBUSY, ACTIVITYERR, INVREQ, IOERR, LOCKED

Description

CHECK ACTIVITY returns the completion status of a BTS activity. Typically, it is used to check the success of a previous RUN ACTIVITY or LINK ACTIVITY command. It allows the requestor to discover whether an activity completed successfully, or whether, for example, it needs to be reactivated in order to complete its processing.

CHECK ACTIVITY can be issued:

1. By a parent activity, to check the completion status of one of its children
2. By a program that has acquired an activity by means of an ACQUIRE ACTIVITYID command.

It can be used to check descendant (not root) activities:

- That have completed
- That have not completed
- That were requested to run asynchronously
- That were requested to run synchronously.

The RESP and RESP2 options on this command reflect whether the command is understood by CICS—for example, ACTIVITYERR occurs if the child named on the ACTIVITY option has not been defined to the parent.

The COMPSTATUS option returns a CVDA value indicating the completion status of the activity—for example, NORMAL is returned if the activity has successfully completed all its processing steps, while INCOMPLETE is returned if it has returned from an activation but needs to be reattached in order to complete its processing.

If this command is issued by a parent activity in respect of one of its children, and the child has completed, on return from the command CICS deletes the child's completion event from the parent's event pool.

For further guidance on the use of the CHECK ACTIVITY command, see Dealing with BTS errors and response codes .

Options

ABCODE(data-area)
returns, if the activity terminated abnormally, the 4-character abend code.

ABPROGRAM(data-area)
returns, if the activity terminated abnormally, the 8-character name of the program that was in control at the time of the abend.

ACQACTIVITY
specifies that the activity to be checked is the one that the current unit of work has acquired by means of an ACQUIRE ACTIVITYID command.

ACTIVITY(data-value)
specifies the name (1–16 characters) of the activity to be checked.

Use this option to check the state of a child of the current activity.

COMPSTATUS(cvda)
indicates the completion status of the activity. CVDA values are:

ABEND
The program that implements the activity abended. Any children of the activity have been canceled.

The activity's completion event is deleted from the parent's event pool.

FORCED
The activity was forced to complete—for example, it was canceled with a CANCEL ACTIVITY command.

The activity's completion event is deleted from the parent's event pool.

INCOMPLETE
The named activity is incomplete. This could mean:

- That it has not yet been run
- That it has returned from one or more activations but needs to be reattached in order to complete all its processing steps
- That it is currently active.

The activity's completion event is **not** deleted from the parent's event pool.

NORMAL
The named activity completed successfully.

The activity's completion event is deleted from the parent's event pool.

MODE(cvda)
indicates the processing state of the activity. CVDA values are:

ACTIVE
An activation of the activity is running.

CANCELLING
CICS is waiting to cancel the activity. A CANCEL ACTIVITY command has been issued, but CICS cannot cancel the activity immediately because one or more of the activity's children are inaccessible.

COMPLETE
The activity has completed.

DORMANT
The activity is waiting for an event to fire its next activation.

INITIAL
No RUN or LINK command has yet been issued against the activity; or the activity has been reset by means of a RESET ACTIVITY command.

SUSPSTATUS(cvda)
indicates whether the activity is currently suspended. CVDA values are:

SUSPENDED
The activity is currently suspended. If a reattachment event occurs, it will not be reactivated.

NOTSUSPENDED
The activity is not currently suspended. If a reattachment event occurs, it will be reactivated.

Conditions

107 ACTIVITYBUSY
RESP2 values:

19 The request timed out. It may be that another task using this activity-record has been prevented from ending.

109 ACTIVITYERR
RESP2 values:

8 The activity named in the ACTIVITY option could not be found.

16 INVREQ
RESP2 values:

4 The ACTIVITY option was used to name a child activity, but the command was issued outside the scope of a currently-active activity.

24 The ACQACTIVITY option was used, but the unit of work that issued the request has not acquired an activity.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

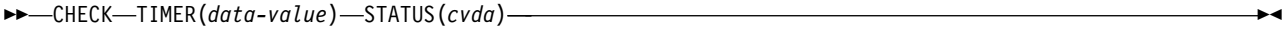
30 An input/output error has occurred on the repository file.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

CHECK TIMER

Check the status of a BTS timer.

CHECK TIMER



Conditions: INVREQ, IOERR, TIMERERR

Description

CHECK TIMER returns the status of a BTS timer. It allows the requestor to discover whether a timer has expired and, if so, whether it expired normally or whether its expiry was forced by means of a FORCE TIMER command.

On return from this command, if the timer has expired its associated event is deleted from the current activity's event pool.

The only timers a program can check are those owned by the current activity.

Options

STATUS(cvda)
indicates the status of the timer. CVDA values are:

EXPIRED
The timer expired normally.

Its associated event is deleted from the current activity's event pool.

FORCED
The timer expired because a FORCE TIMER command was issued against it.

Its associated event is deleted from the current activity's event pool.

UNEXPIRED
The timer has not yet expired.

Its associated event is not deleted from the current activity's event pool.

TIMER(data-value)
specifies the name (1–16 characters) of the timer to be checked.

Conditions

16 INVREQ
RESP2 values:

1 The command was issued outside the scope of a currently-active activity.

17 IOERR
An input/output error has occurred on the repository file.

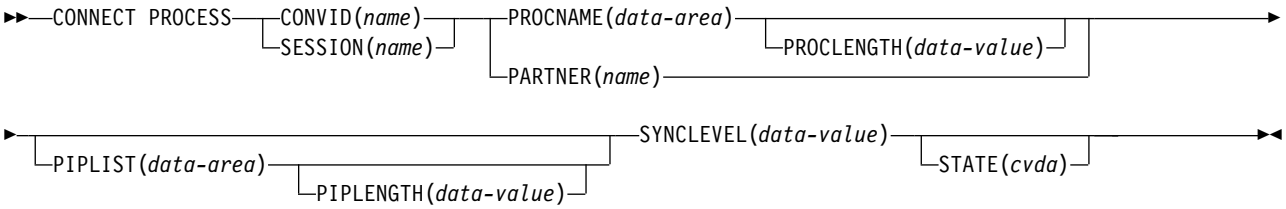
115 TIMERERR
RESP2 values:

13 The timer specified on the TIMER option does not exist.

CONNECT PROCESS

Initiate an APPC mapped conversation.

CONNECT PROCESS (APPC)



Conditions: INVREQ, LENGERR, NOTALLOC, PARTNERIDERR, TERMERR

Description

CONNECT PROCESS allows an application to specify a process name and synchronization level to be passed to CICS and used when the remote partner is attached.

Options

CONVID(*name*)
identifies the conversation to which the command relates. The 4-character name specifies the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB.

For compatibility with earlier releases, SESSION is accepted as a synonym for CONVID. New programs should use CONVID.

PARTNER(*name*)
specifies the name (8 characters) of a set of definitions that includes the name (or extended name) of a remote partner transaction (TPNAME or XTPNAME). You can use this option as an alternative to PROCNAME and PROCLENGTH.

PIPLENGTH(*data-value***)**
specifies the total length (halfword binary value) of the specified process initialization parameter (PIP) list.

PIPLIST(*data-area***)**
specifies the PIP data to be sent to the remote system. The PIP list consists of variable-length records, each containing a single PIP. A PIP starts with a 2-byte inclusive length field (LL), followed by a 2-byte reserved field, and then the parameter data.

PROCLENGTH(*data-value***)**
specifies the length (as a halfword binary value in the range 1–64) of the name specified by the PROCNAME option.

PROCNAME(*data-area***)**
specifies the partner process (that is, the transaction) to be attached in the remote system.

One byte is sufficient to identify a CICS transaction. The APPC architecture allows a range of 1–64 bytes but leaves each product free to set its own maximum. CICS complies by allowing a range of 1–64 bytes. If the remote system is CICS, this option can specify the 4-byte transaction identifier or the TPNAME value given in the relevant TRANSACTION definition. Alternatively, you can examine the full identifier by coding the user exit XZCATT.

No character checking is performed on the TPN by CICS.

For programming information about the user exit XZCATT, see CICS statistics record format .

SESSION(*name***)**
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

SYNCLEVEL(*data-value***)**
specifies the synchronization level (halfword binary value) for the current conversation. The possible values are:

- 0 None
- 1 Confirm

- 2 Syncpoint

Conditions

16 INVREQ

RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

also occurs (RESP2 not set) in any of the following situations:

- A synchronization level other than 0, 1, or 2, has been requested in the SYNCLEVEL option.
- The command is not valid for the terminal or LU in use.
- The command has been used on a conversation that is in use by CPI-Communications or that is an APPC basic conversation. In the latter case, GDS CONNECT PROCESS should have been used.

Default action: terminate the task abnormally.

22 LENGERR

occurs in any of the following situations:

- An out-of-range value is supplied in the PROCLENGTH option.
- The value specified in the PIPLENGTH option is less than 0.
- The value specified in the PIPLENGTH option exceeds the CICS implementation limit of 32 763.
- A PIPLIST length element (LL) has a value less than 4.
- The sum of the length elements (LLs) in the PIPLIST does not equal the value specified by PIPLENGTH.

Default action: terminate the task abnormally.

61 NOTALLOC

occurs if the specified CONVID value does not relate to a conversation owned by the application.

Default action: terminate the task abnormally.

97 PARTNERIDERR

occurs if the name specified in the PARTNER option is not recognized by CICS.

Default action: terminate the task abnormally.

81 TERMERR

occurs for a session-related error. Any action on that conversation other than a FREE causes an ATCV abend.

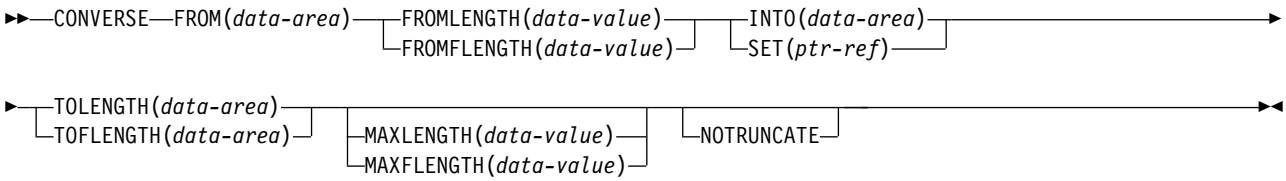
A CANCEL TASK request by a user node error program (NEP) can cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

CONVERSE (default)

Communicate on standard CICS terminal support.

CONVERSE (default)



Conditions: LENGERR

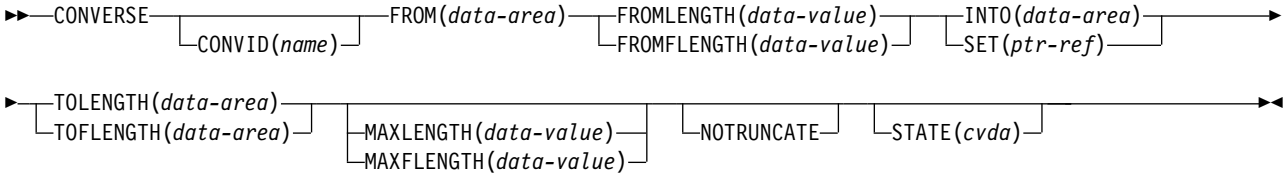
Description

This form of the CONVERSE command is used by all CICS-supported z/OS Communications Server terminals for which the other CONVERSE descriptions are not appropriate.

CONVERSE (APPC)

Communicate on an APPC mapped conversation

CONVERSE (APPC)



Conditions: EOC, INVREQ, LENGERR, NOTALLOC, SIGNAL, TERMERR

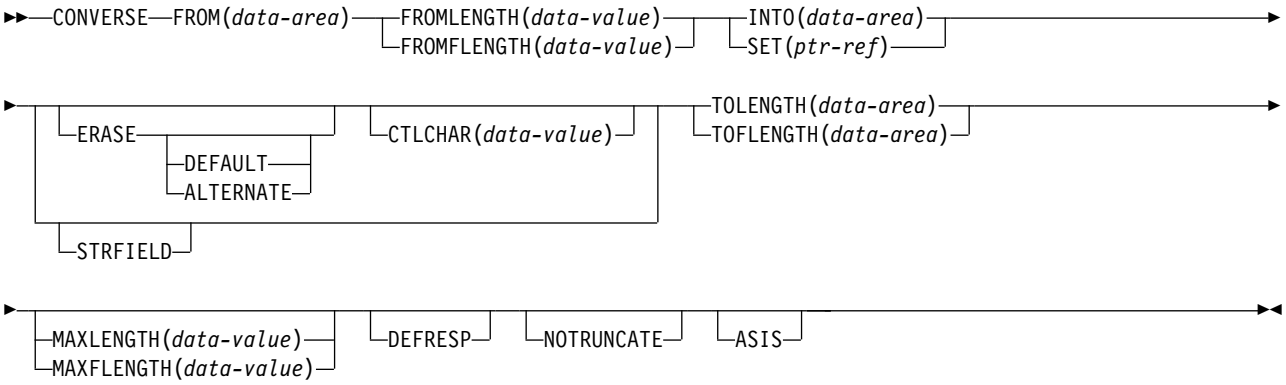
Description

CONVERSE sends, then receives, data on an APPC mapped conversation.

CONVERSE (LUTYPE2/LUTYPE3)

Communicate on a 3270-display logical unit (LUTYPE2) or 3270-printer logical unit (LUTYPE3).

CONVERSE (LUTYPE2/LUTYPE3)



Conditions: EOC, LENGERR, TERMERR

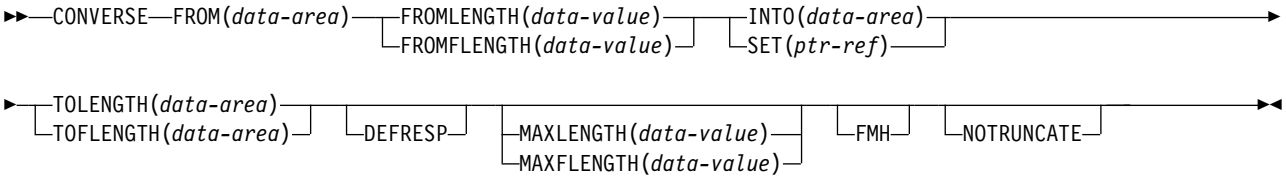
Description

CONVERSE communicates on a 3270-display logical or 3270-printer logical unit.

CONVERSE (LUTYPE4)

Communicate on an LUTYPE4 logical unit.

CONVERSE (LUTYPE4)



Conditions: EOC, EODS, IGREQCD, INBFMH, LENGERR, SIGNAL, TERMERR

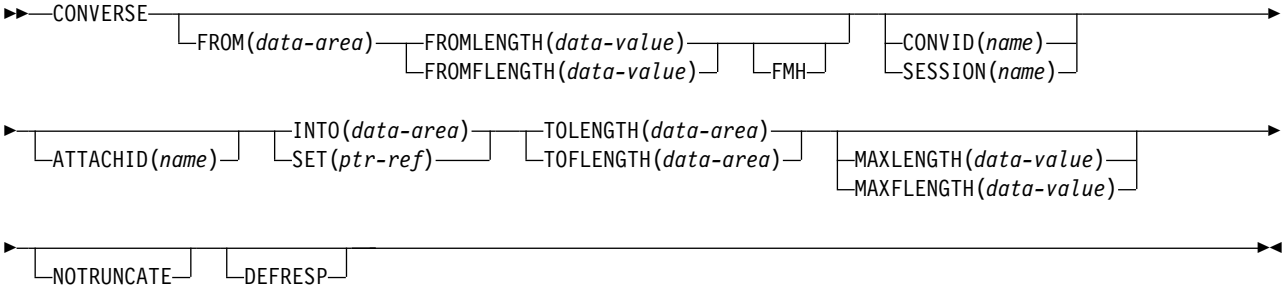
Description

CONVERSE communicates on an LUTYPE4 logical unit.

CONVERSE (LUTYPE6.1)

Communicate on an LUTYPE6.1 logical unit.

CONVERSE (LUTYPE6.1)



Conditions: CBIDERR, EOC, INBFMH, LENGERR, NOTALLOC, SIGNAL, TERMERR

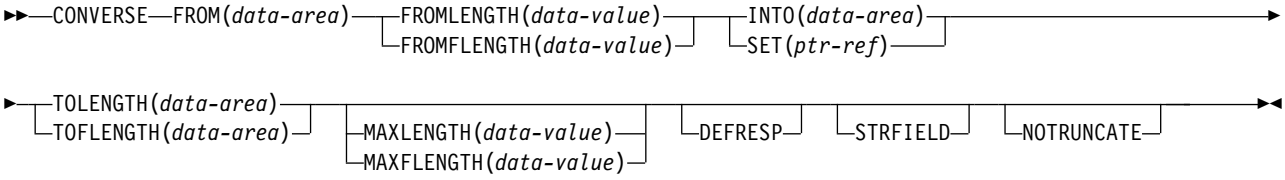
Description

CONVERSE communicates on an LUTYPE6.1 logical unit.

CONVERSE (SCS)

Communicate on a 3270 SCS printer logical unit.

CONVERSE (SCS)



Conditions: LENGERR, TERMERR

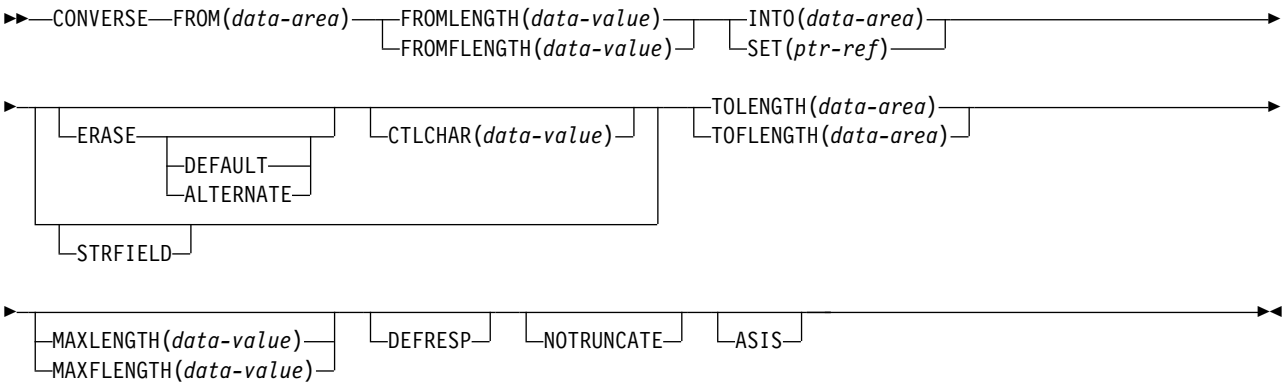
Description

CONVERSE communicates on a 3270 SNA character string (SCS) printer logical unit. The SCS printer logical unit accepts a character string as defined by Systems Network Architecture (SNA). Some devices connected under SNA can send a signal that can be detected by the HANDLE CONDITION SIGNAL command, which in turn can invoke an appropriate handling routine. If necessary, a WAIT SIGNAL command can be used to make the application program wait for the signal. The PA keys on a 3287 can be used in this way, or with a RECEIVE command.

CONVERSE (3270 logical)

Communicate on a 3270 logical unit.

CONVERSE (3270 logical)



Conditions: INVREQ, LENGERR, TERMERR

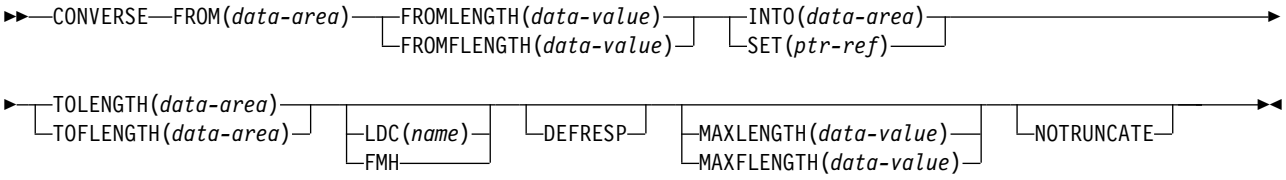
Description

CONVERSE communicates on a 3270 logical unit.

CONVERSE (3600-3601)

Communicate on a 3600 (3601) logical unit.

CONVERSE (3600-3601)



Conditions: EOC, EODS, INBFMH, LENGERR, SIGNAL, TERMERR

Description

CONVERSE communicates on a 3600 logical unit. This form of the CONVERSE command also applies to the 4700 and the 3630 plant communication system.

A logical device code (LDC) is a code that can be included in an outbound Function Management Header (FMH) to specify the disposition of the data (for example, to which subsystem terminal it should be sent). Each code can be represented by a unique LDC mnemonic.

The installation can specify up to 256 2-character mnemonics for each TCTTE, and two or more TCTTEs can share a list of these mnemonics. A numeric value (0 through 255) corresponds to each LDC mnemonic for each TCTTE.

A 3600 device and a logical page size are also associated with an LDC. LDC or *LDC value* is used in this information to refer to the code specified by the user; *LDC mnemonic* refers to the 2-character symbol that represents the LDC numeric value.

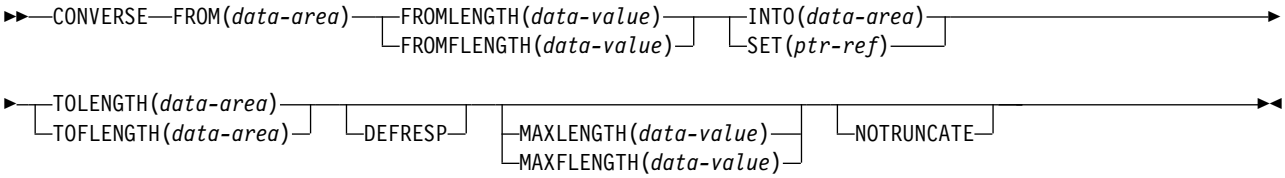
When the LDC option is specified in the CONVERSE command, the numeric value associated with the mnemonic for the particular TCTTE is inserted in the FMH. This value is chosen by the installation, and is interpreted by the 3601 application program.

On output, the FMH can be built by the application program or by CICS. If your program supplies the FMH, you place it at the front of your output data and specify the FMH option on your CONVERSE command. If you omit the FMH option, CICS will provide an FMH but you must reserve the first three bytes of the message for CICS to complete.

CONVERSE (3600-3614)

Communicate on a 3600 (3614) logical unit.

CONVERSE (3600-3614)



Conditions: LENGERR, TERMERR

Description

CONVERSE communicates on a 3600 logical unit.

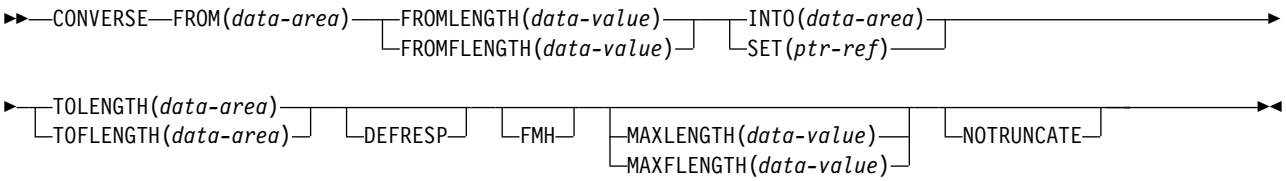
The data stream and communication format used between a CICS application program and a 3614 is determined by the 3614. The application program is therefore device_dependent when handling 3614 communication.

For further information about designing 3614 application programs for CICS, refer to the *IBM 4700/3600/3630 Guide*.

CONVERSE (3650 interpreter)

Communicate on a 3650 interpreter logical unit.

CONVERSE (3650 interpreter)



Conditions: EOC, EODS, INBFMH, LENGERR, TERMERR

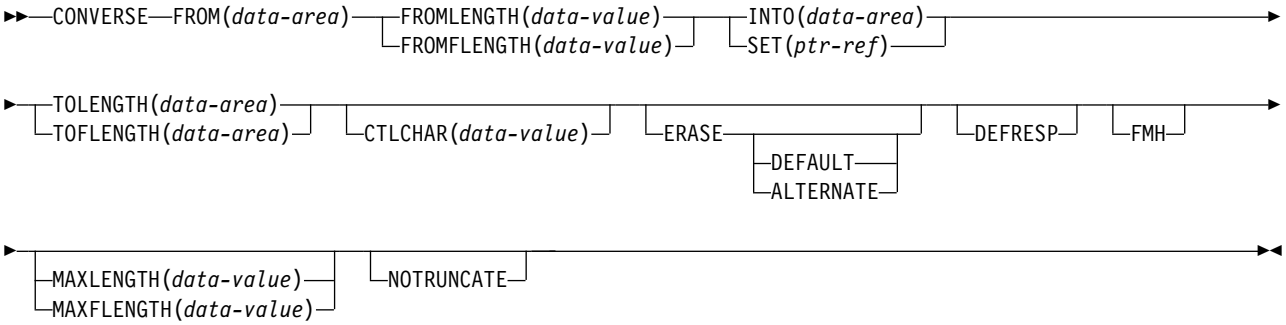
Description

CONVERSE communicates on a 3650 interpreter logical unit.

CONVERSE (3650-3270)

Communicate on a 3650 host conversational (3270) logical unit.

CONVERSE (3650-3270)



Conditions: LENGERR, TERMERR

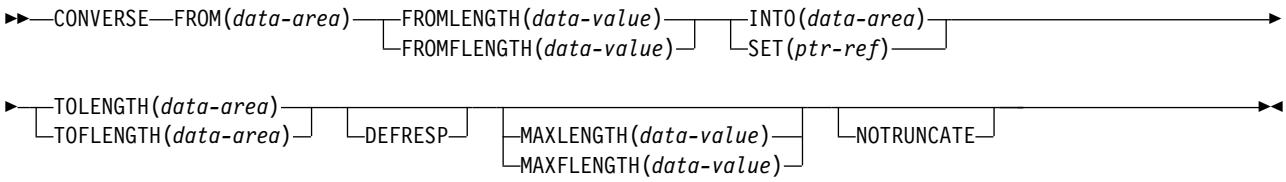
Description

CONVERSE communicates on a 3650 host conversational logical unit.

CONVERSE (3650-3653)

Communicate on a 3650 host conversational (3653) logical unit.

CONVERSE (3650-3653)



Conditions: EOC, LENGERR, TERMERR

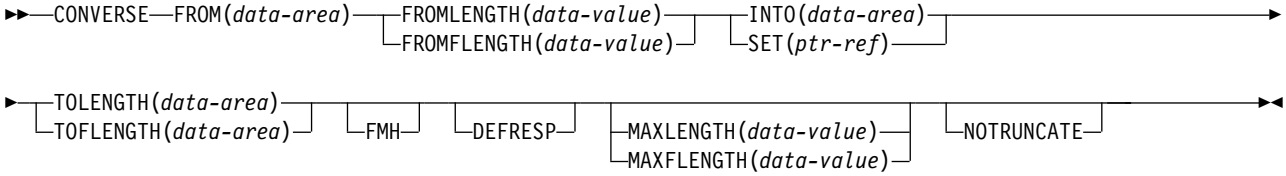
Description

CONVERSE communicates on a 3650 host conversational logical unit.

CONVERSE (3650-3680)

Communicate on a 3650 host command processor (3680) logical unit.

CONVERSE (3650-3680)



Conditions: LENGERR, TERMERR

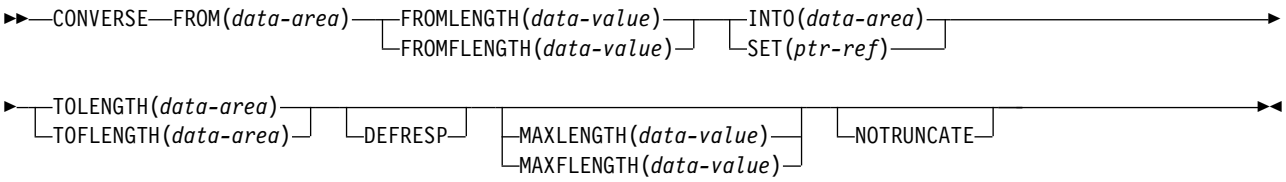
Description

CONVERSE communicates on a 3650 host command processor logical unit.

CONVERSE (3767)

Communicate on a 3767 interactive logical unit.

CONVERSE (3767)



Conditions: EOC, LENGERR, SIGNAL, TERMERR

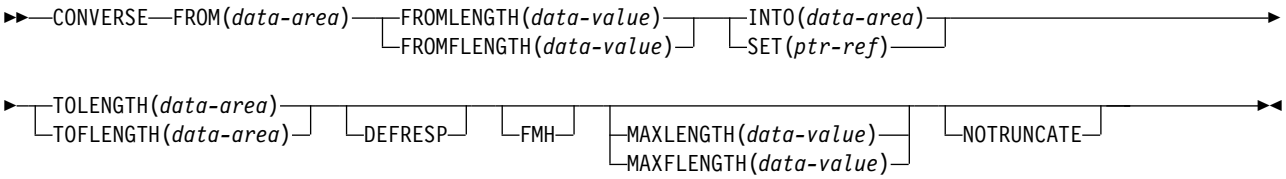
Description

CONVERSE communicates on a 3767 interactive logical unit. This command also applies to the 3770 interactive logical unit.

CONVERSE (3770)

Communicate on a 3770 batch logical unit.

CONVERSE (3770)



Conditions: EOC, EODS, INBFMH, LENGERR, SIGNAL, TERMERR

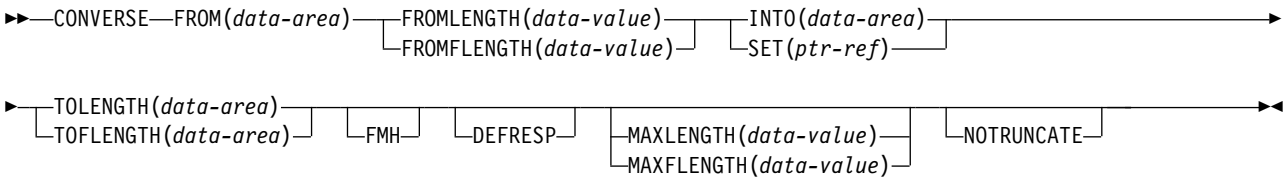
Description

CONVERSE communicates on a 3770 batch logical unit.

CONVERSE (3790 full-function or inquiry)

Communicate on a 3790 full-function or inquiry logical unit.

CONVERSE (3790 full-function or inquiry)



Conditions: EOC, EODS, INBFMH, LENGERR, SIGNAL, TERMERR

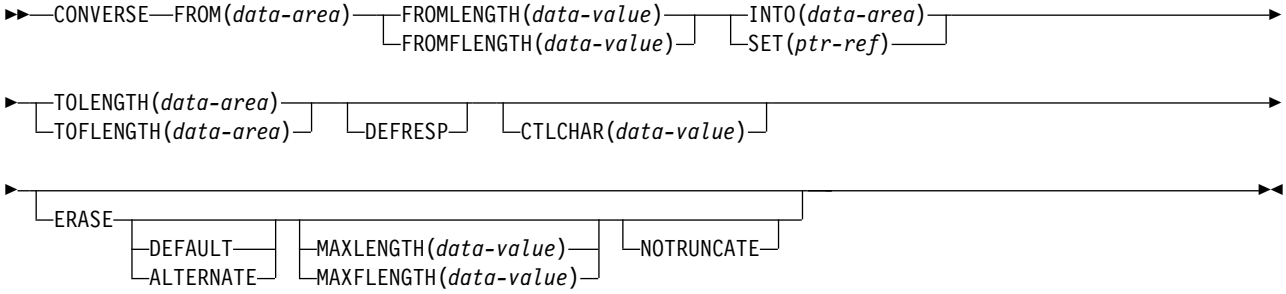
Description

CONVERSE communicates on a 3790 full-function or inquiry logical unit.

CONVERSE (3790 3270-display)

Communicate on a 3790 (3270-display) logical unit.

CONVERSE (3790 3270-display)



Conditions: LENGERR, TERMERR

Description

CONVERSE communicates on a 3790 logical unit.

CONVERSE: z/OS Communications Server options

Common options used on the CONVERSE command (z/OS Communications Server).

Options

ALTERNATE
sets the terminal to use the ALTERNATE screen size.

ASIS
indicates that output is to be sent in transparent mode (with no recognition of control characters and accepting any of the 256 possible combinations of eight bits as valid transmittable data).

Note: If you are using a katakana terminal, you might see some messages containing mixed English and katakana characters. That is because katakana terminals cannot display mixed-case output. Uppercase characters in the data stream appear as uppercase English characters, but lowercase characters appear as katakana characters. If this happens, ask your system programmer to specify MSGCASE=UPPER in the system initialization parameters so that messages contain uppercase characters only. This note applies to any command that is used to receive katakana characters, not just to CONVERSE commands.

ATTACHID(*name***)**
specifies that an attach header (created by a BUILD ATTACH command) is to precede, and be concatenated with, the user data supplied in the FROM option. “name” (1–8 characters) identifies the attach header control block to be used in the local task.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

For compatibility with earlier releases, SESSION is accepted as a synonym for CONVID. New programs should use CONVID.

If the option is omitted, the principal facility for the task is used by default.

CTLCHAR(*data-value***)**
specifies a 1-byte write control character (WCC) that controls the CONVERSE command. A COBOL user must specify a data area containing this character.

If the option is omitted, all modified data tags are reset to zero, and the keyboard is restored.

DEFAULT
sets the terminal to use the DEFAULT screen size.

DEFRESP
indicates that a definite response is required when the output operation has been completed.

ERASE
specifies that the screen printer buffer or partition is to be erased and the cursor returned to the upper left corner of the screen. (This option applies only to the 3270, or 8775, and to the 3604 Keyboard Display.)

The first output operation in any transaction, or in a series of pseudoconversational transactions, should always specify ERASE. For transactions attached to 3270 screens or printers, unless explicitly overridden by the DEFAULT or ALTERNATE option, this also ensures that the correct screen size is selected, as defined for the transaction by the SCRNSIZE option in the RDO PROFILE definition.

FMH
specifies that a function management header has been included in the data to be written. If the ATTACHID option is specified as well, the concatenated FMH flag is set in the attach FMH.

The use of FMH is optional and is not supported for all terminal types. If not supplied, CICS takes no action, except for 3600/4700 terminals, where an FMH is mandatory. In this case, if FMH is not specified, CICS supplies one and places it in the first 3 bytes of the message, which you must reserve for this purpose.

FROM(*data-area***)**
specifies the data to be written to the terminal or logical unit, or sent to the partner transaction. This option may, when relevant, be omitted if ATTACHID is specified.

FROMFLENGTH(*data-value***)**
is a fullword alternative to FROMLENGTH.

FROMLENGTH(*data-value***)**
specifies the length, as a halfword binary value, of the data. For a description of a safe upper limit, see "LENGTH options in CICS commands" on page 10.

INTO(*data-area***)**
specifies the receiving field for the data read from the terminal or logical unit, or the application target data area into which data is to be received from the application program connected to the other end of the current conversation.

LDC(*name***)**
specifies the 2-character mnemonic used to determine the appropriate logical device code (LDC) numeric value. The mnemonic identifies an LDC entry defined by a DFHTCT TYPE=LDC macro.

MAXFLENGTH(*data-value***)**
is a fullword alternative to MAXLENGTH.

MAXLENGTH(*data-value***)**
specifies the maximum amount (halfword binary value) of data that CICS is to recover in response to a CONVERSE (default) command. If INTO is specified, MAXLENGTH overrides the use of TOLENGTH as an input to CICS. If SET is specified, MAXLENGTH provides a way for the program to limit the amount of data it receives at one time.

If the value specified is less than zero, zero is assumed.

If the length of data exceeds the value specified and the NOTRUNCATE option is not present, the data is truncated to that value and the LENGERR condition occurs. The data area specified in the TOLENGTH option is set to the original length of data.

If the length of data exceeds the value specified and the NOTRUNCATE option is present, CICS retains the remaining data and uses it to satisfy subsequent RECEIVE commands. The data area specified in the TOLENGTH option is set to the length of data returned.

If no argument is coded for MAXLENGTH, CICS defaults to TOLENGTH.

NOTRUNCATE
specifies that, when the data available exceeds the length requested, the remaining data is not to be discarded but is to be retained for retrieval by subsequent RECEIVE commands.

SESSION(*name***)**
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used. If both this option and CONVID are omitted, the principal facility for the task is used.

SET(*ptr-ref***)**
specifies the pointer reference to be set to the address of the data read from the terminal. pointer reference, unless changed by other commands or statements, is valid until the next CONVERSE (default) command or the end of task.

If DATALOCATION(ANY) is associated with the application program, the address of the data can be above or below the 16MB line.

If DATALOCATION(BELOW) is associated with the application program, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

STRFIELD
specifies that the data area specified in the FROM option contains structured fields. If this option is specified, the contents of all structured fields must be handled by the application program. The CONVERSE command must be used if the data area contains a read partition structured field. (Structured fields are described in the *CICS 3270 Data Stream Device Guide*.)

CTLCHAR and ERASE are mutually exclusive with STRFIELD, and their use with STRFIELD generates an error message.

TOFLENGTH(*data-area***)**
is a fullword alternative to TOLENGTH.

TOLENGTH(*data-area***)**
specifies the length (halfword binary value) of the data to be received. If you specify INTO, but omit MAXLENGTH, “data-area” specifies the maximum length that the program accepts. If the value is less than zero, zero is assumed.

If the length of the data exceeds the value specified, but NOTRUNCATE is omitted, the data is truncated to that value, and the LENGERR condition occurs. When the data is received, the data area is set to the length of the data.

For a description of a safe upper limit, see "LENGTH options in CICS commands" on page 10.

Conditions

Some of the following conditions can occur in combination with others. CICS checks for these conditions in the following order:

1. EODS
2. INBFMH
3. EOC

If more than one occurs, only the first is passed to the application program. EIBRCODE, however, is set to indicate all the conditions that occurred.

62 CBIDERR
occurs if the requested attach header control block named in ATTACHID cannot be found.

Default action: terminate the task abnormally.

06 EOC
occurs when a request/response unit (RU) is received with the end-of-chain indicator set. Field EIBEOC also contains this indicator.

Default action: ignore the condition.

05 EODS
occurs when an end-of-data-set indicator is received.

Default action: terminate the task abnormally.

57 IGREQCD
occurs when an attempt is made to execute a CONVERSE command after a SIGNAL data-flow control command with a request change direction (RCD) code has been received from an LUTYPE4 logical unit.

Default action: terminate the task abnormally.

07 INBFMH
occurs if a request/response unit (RU) contains a function management header (FMH). Field EIBFMH contains this indicator and it should be used in preference to INBFMH. The IGNORE CONDITION command can be used to ignore the condition.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function shipping session (its principal facility)

also occurs (RESP2 not set) in any of the following situations:

- The command is used on a conversation that is in use by CPI Communications, or that is an APPC basic conversation. In the latter case, the application should have issued a GDS SEND INVITE followed by a GDS RECEIVE.

Default action: terminate the task abnormally.

22 LENGERR
occurs in any of the following situations:

- Data received is discarded by CICS because its length exceeds the maximum that the program accepts (see TOLENGTH and MAXLENGTH options), and the NOTRUNCATE option is not specified.
- An out-of-range value is supplied in one of the options, FROMLENGTH, FROMFLENGTH, MAXLENGTH, MAXFLENGTH, TOLENGTH, or TOFLENGTH.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application, or does not relate to a conversation owned by the application.

Default action: terminate the task abnormally.

24 SIGNAL
occurs when an inbound SIGNAL data-flow control command is received from a logical unit or session, or the partner transaction. EIBSIG is always set when an inbound signal is received.

Default action: ignore the condition.

81 TERMERR
occurs for a terminal or session-related error. Any action on that conversation other than a FREE causes an ATCV abend.

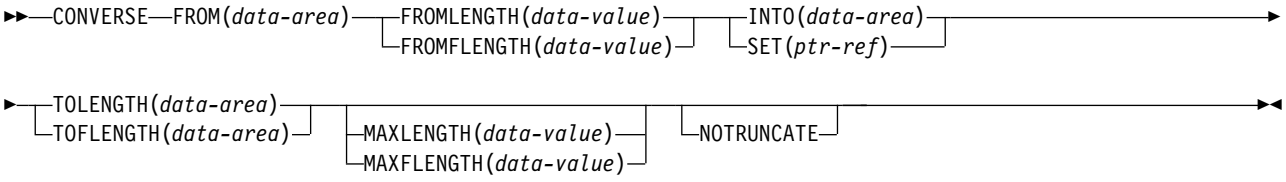
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

CONVERSE (non-z/OS Communications Server default)

Communicate on standard CICS terminal support.

CONVERSE (default)



Conditions: LENGERR

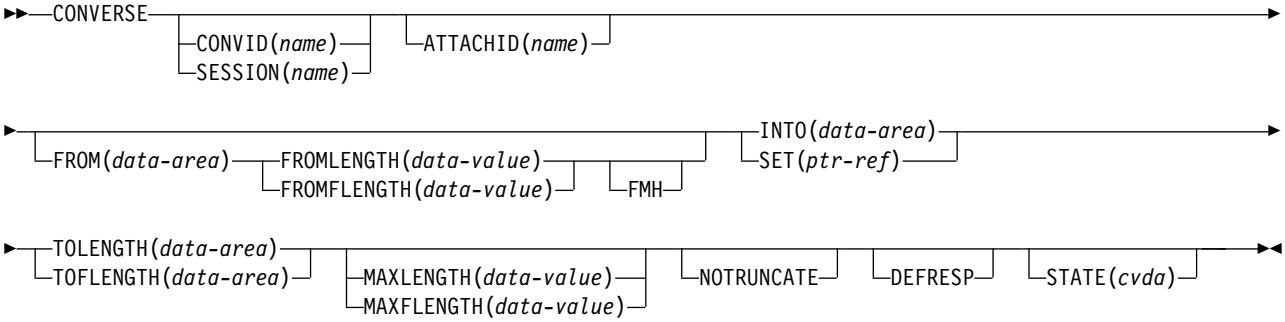
Description

This form of the CONVERSE command is used by all CICS-supported terminals for which the other CONVERSE descriptions are not appropriate.

CONVERSE (MRO)

Communicate on an MRO session.

CONVERSE (MRO)



Conditions: CBIDERR, EOC, INBFMH, LENGERR, NOTALLOC, TERMERR

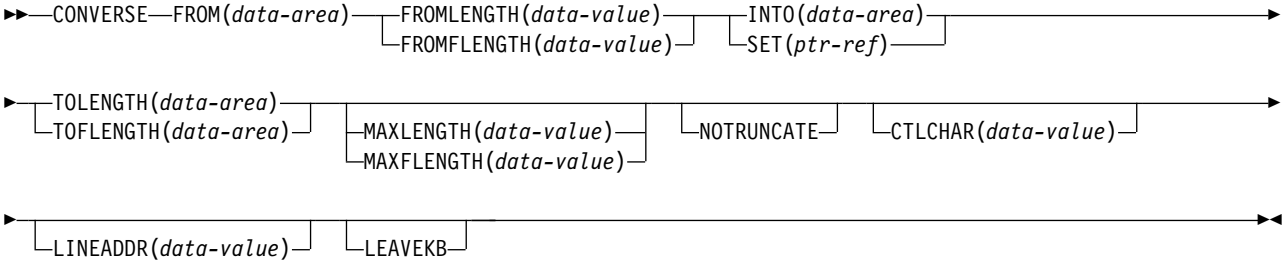
Description

CONVERSE communicates on an MRO session. For more information about MRO and IRC, see Introduction to CICS intercommunication.

CONVERSE (2260)

Communicate on a 2260 or 2265 display station.

CONVERSE (2260)



Condition: LENGERR

Description

CONVERSE communicates on a 2260 or 2265 display station.

CONVERSE: non-z/OS Communications Server options

Common options used on the CONVERSE command (non-z/OS Communications Server).

Options

ALTERNATE
set the terminal to use the ALTERNATE screen size.

ASIS
indicates that output is to be sent in transparent mode (with no recognition of control characters and accepting any of the 256 possible combinations of eight bits as valid transmittable data).

Note: If you are using a katakana terminal, you might see some messages containing mixed English and katakana characters. That is because katakana terminals cannot display mixed-case output. Uppercase characters in the data stream appear as uppercase English characters, but lowercase characters appear as katakana characters. If this happens, ask your system programmer to specify MSGCASE=UPPER in the system initialization parameters so that messages contain uppercase characters only.

ATTACHID(*name***)**
specifies that an attach header (created by a BUILD ATTACH command) is to precede, and be concatenated with, the user data supplied in the FROM option. “name” (1–8 characters) identifies the attach header control block to be used in the local task.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

CTLCHAR(*data-value***)**
specifies a 1-byte write control character (WCC) that controls the CONVERSE command. (The WCC is documented in the *IBM 3270 Data Stream Programmer's Reference* manual.) A COBOL user must specify a data area containing this character. If the option is omitted, all modified data tags are reset to zero and the keyboard is restored.

DEFAULT
set the terminal to use the DEFAULT screen size.

DEFRESP
indicates that a definite response is required when the output operation has been completed.

ERASE
specifies that the screen printer buffer or partition is to be erased and the cursor returned to the upper left corner of the screen. (This option applies only to the 3270, or 8775, and to the 3604 Keyboard Display.)

The first output operation in any transaction, or in a series of pseudoconversational transactions, should always specify ERASE. For transactions attached to 3270 screens or printers, unless explicitly overridden by the DEFAULT or ALTERNATE option, this also ensures that the correct screen size is selected, as defined for the transaction by the SCRNSIZE option in the RDO PROFILE definition.

FMH
specifies that a function management header has been included in the data to be written. If the ATTACHID option is specified as well, the concatenated FMH flag is set in the attach FMH.

FROM(*data-area***)**
specifies the data to be written to the terminal or logical unit, or sent to the partner transaction. This option may, when relevant, be omitted if ATTACHID is specified.

FROMFLENGTH(*data-value***)**
is a fullword alternative to FROMLENGTH.

FROMLENGTH(*data-value***)**
specifies the length, as a halfword binary value, of the data to be written. If you use this option, you must also specify FROM. For a description of a safe upper limit, see "LENGTH options in CICS commands" on page 10.

INTO(*data-area***)**
specifies the receiving field for the data read from the logical unit or terminal.

LEAVEKB
specifies that the keyboard is to remain locked at the completion of the data transfer.

LINEADDR(*data-value***)**
specifies that the writing is to begin on a specific line of a 2260/2265 screen. The data value is a halfword binary value in the range 1 through 12 for a 2260, or 1 through 15 for a 2265.

MAXFLENGTH(*data-value***)**
is a fullword alternative to MAXLENGTH.

MAXLENGTH(*data-value***)**
specifies the maximum amount (halfword binary value) of data that CICS is to recover in response to a CONVERSE command. If INTO is specified, MAXLENGTH overrides the use of TOLENGTH as an input to CICS. If SET is specified, MAXLENGTH provides a way for the program to limit the amount of data it receives at one time.

If the value specified is less than zero, zero is assumed.

If the length of data exceeds the value specified and the NOTRUNCATE option is not present, the data is truncated to that value and the LENGERR condition occurs. The data area specified in the TOLENGTH option is set to the original length of data.

If the length of data exceeds the value specified and the NOTRUNCATE option is present, CICS retains the remaining data and uses it to satisfy subsequent RECEIVE commands. The data area specified in the TOLENGTH option is set to the length of data returned.

If no argument is coded for MAXLENGTH, CICS defaults to TOLENGTH.

NOTRUNCATE
specifies that, when the data available exceeds the length requested, the remaining data is not to be discarded but retained for retrieval by subsequent RECEIVE commands.

PSEUDOBIN
specifies that the data being read and written is to be translated from System/7 pseudobinary representation to hexadecimal.

SESSION(*name***)**
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used. If both this option and CONVID are omitted, the principal facility for the task is used.

SET(*ptr-ref***)**
specifies a pointer reference to be set to the address of data received from the conversation partner in an MRO conversation. The pointer reference, unless changed by other commands or statements, is valid until the next CONVERSE (MRO) command or the end of task.

If DATALOCATION(ANY) is associated with the application program, the address of the data can be above or below the 16MB line.

If DATALOCATION(BELOW) is associated with the application program, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

STATE(*cvda***)**
gets the state of the transaction program. The cvda values returned by CICS are:

- ALLOCATED
- FREE
- PENDFREE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

STRFIELD
specifies that the data area specified in the FROM option contains structured fields. If this option is specified, the contents of all structured fields must be handled by the application program. The CONVERSE command, rather than a SEND command, must be used if the data area contains a read partition structured field. (Structured fields are described in the *CICS 3270 Data Stream Device Guide*.)

CTLCHAR and ERASE are mutually exclusive with STRFIELD, and their use with STRFIELD generates an error message.

TOFLENGTH(*data-area***)**
is a fullword alternative to TOLENGTH.

TOLENGTH(*data-area***)**
specifies the length, as a halfword binary value, of the data to be received. If you specify INTO, but omit MAXLENGTH, “data-area” specifies the maximum length that the program accepts. If the value is less than zero, zero is assumed.

If the length of the data exceeds the value specified, but NOTRUNCATE is omitted, the data is truncated to that value, and the LENGERR condition occurs. When the data is received, the data area is set to the length of the data.

For a description of a safe upper limit, see “LENGTH options in CICS commands” on page 10.

Conditions

Some of the following conditions can occur in combination with others. CICS checks for these conditions in the following order:

1. INBFMH
2. EOC

If more than one occurs, only the first is passed to the application program. EIBRCODE, however, is set to indicate all the conditions that occurred.

62 CBIDERR
occurs if the requested attach header control block named in ATTACHID cannot be found.

Default action: terminate the task abnormally.

06 EOC
occurs when a request/response unit (RU) is received with the end-of-chain indicator set. Field EIBEOC also contains this indicator.

Default action: ignore the condition.

04 EOF
occurs when an end-of-file indicator is received.

Default action: terminate the task abnormally.

07 INBFMH
occurs if a request/response unit (RU) contains a function management header (FMH). Field EIBFMH contains this indicator and it should be used in preference to INBFMH. The IGNORE CONDITION command can be used to ignore the condition.

Default action: terminate the task abnormally.

22 LENGERR
occurs in any of the following situations:

- Data is discarded by CICS because its length exceeds the maximum that the program accepts and the NOTRUNCATE option is not specified.
- An out-of-range value is supplied in the FROMLENGTH option.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

02 RDATT
occurs if the "receive" part of the conversation is terminated by the attention (ATTN) key rather than the return key.

Default action: ignore the condition.

81 TERMERR
occurs for a session-related error.

A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

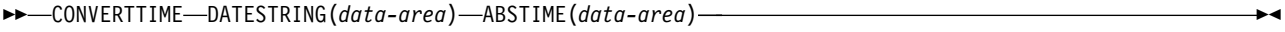
03 WRBRK
occurs if the "send" part of the conversation is terminated by the attention (ATTN) key rather than the return key.

Default action: ignore the condition.

CONVERTTIME

Converts an architected date and time stamp string to the ABSTIME format.

CONVERTTIME



Conditions: INVREQ

This command is threadsafe.

Description

CONVERTTIME analyzes four different date and time stamp formats that are commonly used on the Internet, and converts them to the ABSTIME (absolute date and time) format, in local time.

ABSTIME format gives the time, in packed decimal, since 00:00 on 1 January 1900. The time is given in milliseconds, and it is always truncated, never rounded up. The FORMATTIME command can be used to change this data into other formats.

Here are the architected date and time stamp string formats recognized by the CONVERTTIME command:

RFC 1123 format
The preferred standard format for date and time stamps for the HTTP protocol, as specified in RFC 1123. An example of a date and time stamp in this format is "Tue, 01 Apr 2003 10:01:02 +0000".

RFC 3339 format
The XML dateTime datatype, specified in RFC 3339, which is taken from the ISO 8601 standard. An example of a date and time stamp in this format is "2003-04-01T10:01:02.498Z". Date and time stamps in this format are in UTC (Coordinated Universal Time), with the time zone offset (-12:00 to +12:00) indicated at the end of the date and time stamp, or the letter Z for a zero offset (+00:00). The decimal fraction of a second that is shown in the example is optional.

RFC 850 format
An older date and time stamp format for the Internet, specified in RFC 850. An example of a date and time stamp in this format is "Tuesday, 01-Apr-03 10:01:02 GMT".

Important: Because the year has only two digits in this format, CICS uses the assumption that the years are in the range 1970 to 2069. In the example above, CICS assumes that the date of the document is 1 April 2003. Given the date and time stamp "Thursday, 13-Feb-98 15:30:00 GMT", CICS assumes that the date of the document is 13 February 1998. Be aware of

this behavior when coding your application, if you think that you might receive date and time stamps in this format.

ASCtime format
A date and time stamp format produced by the C ASCtime function. An example of a date and time stamp in this format is "Tue Apr 1 10:01:02 2003".

Options

DATESTRING*(data-area)*
Specifies a 64-character data area to contain the architected date and time stamp string. You can supply a string in any of the formats recognized by the command. If the string is less than 64 characters long, pad it with blanks or nulls. You do not have to specify the format of the data in the DATESTRING option, because CICS automatically reads the data to determine whether it is a supported format. The date and time are converted to local time for the ABSTIME that is returned.

ABSTIME*(data-area)*
Specifies a data-area to receive the converted date and time stamp in ABSTIME format. If the date and time stamp is not in a recognized format, the ABSTIME is returned as zero.

Conditions

16 INVREQ
RESP2 values are:

1 The format of the date and time stamp string is not recognized as any of the formats supported by this command. This error can be caused by a date and time stamp string that is in a supported format but contains formatting errors, such as a year value that has more or less than the correct number of digits for the format, or an item that should be numeric but is not numeric.

2 Invalid time.

3 Invalid month.

4 Invalid year (includes years before 1900).

5 Invalid day name.

6 Invalid day number for month and year specified.

7 GMT was not stated (required for RFC 850 formats).

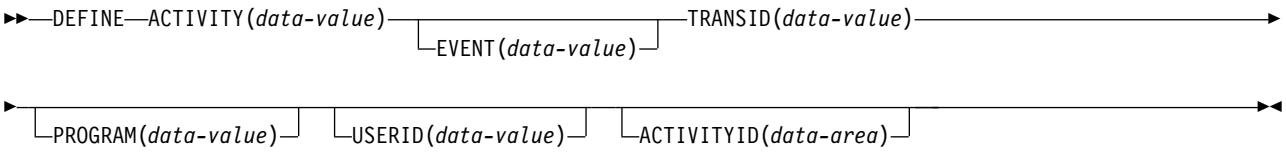
8 Invalid fraction of a second.

9 Invalid time zone offset value.

DEFINE ACTIVITY

Define a CICS business transaction services activity.

DEFINE ACTIVITY



Conditions: ACTIVITYERR, EVENTERR, INVREQ, IOERR, NOTAUTH, TRANSIDERR

Description

DEFINE ACTIVITY defines an activity to CICS business transaction services. It is used to add a child activity to the current activity.

The name of the program used in the execution of the new activity is taken either from the PROGRAM option, or, if PROGRAM is not specified, from the transaction definition pointed to by the TRANSID option.

The transaction attributes specified on the TRANSID and USERID options take effect when the activity is activated by a RUN command, but *not* if it is activated by a LINK command—see "Context-switching" on page 475.

BTS does not commit the addition of the activity until the requesting transaction has taken a successful syncpoint.

Options

ACTIVITY(data-value)
specifies the name (1–16 characters) of the new activity. The name must not be the name of another child activity of the activity that issues the DEFINE command.

The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = ¬ , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

ACTIVITYID(data-area)
returns the 52-character identifier assigned by CICS to the newly-defined activity. This identifier is unique across the sysplex.

EVENT(data-value)
specifies the name (1–16 characters) of the completion event for the activity. The completion event is sent to the activity's parent when the activity completes.

If EVENT is not specified, the completion event is given the same name as the activity itself.

The acceptable characters are A-Z a-z 0-9 $ @ # . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

PROGRAM(data-value)
specifies the name (1–8 characters) of the program for the activity being defined. If no program is specified, the name is taken from the TRANSID definition.

TRANSID(data-value)
specifies the name (1–4 characters) of the transaction under which the activity is to run, when it is activated by a RUN command.

Note: If the activity is activated by a LINK command, it is run under the TRANSID of the transaction that issues the LINK.

The transaction must be defined in the CICS region in which the process is running.

USERID(data-value)
specifies the userid (1–8 characters) under whose authority the activity is to run, when it is activated by a RUN command.

Note: If the activity is activated by a LINK command, it is run under the userid of the transaction that issues the LINK.

The value of this field is known as the *defined userid*.

If you omit USERID, the defined userid defaults to the userid under which the transaction that issues the DEFINE command is running—we can call this the *command userid*.

If USERID is specified, CICS performs (at define time) a surrogate security check to verify that the command userid is authorized to use the defined userid. Thus, if you specify USERID, you must authorize the command userid as a surrogate user of the defined userid.

Conditions

109 ACTIVITYERR
RESP2 values:

3 The name specified on the ACTIVITY option has already been used to name another child of the current activity.

111 EVENTERR
RESP2 values:

7 The completion event specified on the EVENT option has already been defined to the current activity's event pool.

16 INVREQ
RESP2 values:

4 The DEFINE ACTIVITY command was issued outside the scope of a currently-active activity.

17 The activity name specified on the ACTIVITY option, or the event name specified on the EVENT option, is invalid.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access the file associated with the BTS repository data set on which details of the activity are to be stored.

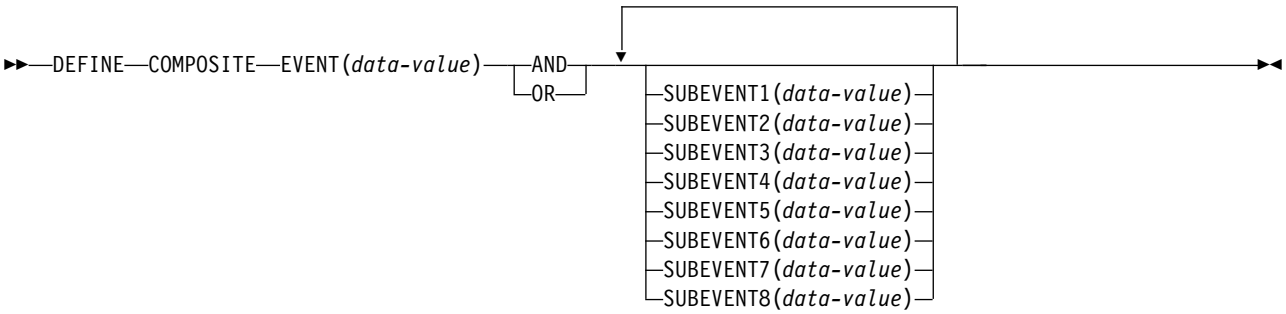
102 The user associated with the issuing task is not authorized as a surrogate of the defined userid specified on the USERID option.

28 TRANSIDERR
The transaction identifier specified on the TRANSID option is not defined to CICS.

DEFINE COMPOSITE EVENT

Define a BTS composite event.

DEFINE COMPOSITE EVENT



Conditions: EVENTERR, INVREQ

Description

DEFINE COMPOSITE EVENT defines a composite event to BTS. A composite event is formed from zero or more atomic events known as sub-events.

DEFINE COMPOSITE EVENT defines a *predicate*, which is a logical expression involving sub-events. At all times, the composite event's fire status (FIRED or NOTFIRED) reflects the value of the predicate. When the predicate becomes true, the composite event fires; when it becomes false, the composite's fire status reverts to NOTFIRED.

The logical operator that is applied to the sub-events in the composite event's predicate is one of the Boolean operators AND or OR. *AND and OR cannot both be used*.

You can specify up to 8 sub-events to be added to the composite event when the composite is created. If you do not specify any sub-events, the composite event is defined as “empty”—that is, as containing no sub-events.

To add sub-events to a composite event after the composite has been defined, use the ADD SUBEVENT command. There is no limit to the number of sub-events that you can add using ADD SUBEVENT.

Note: The following *cannot* be added as sub-events to a composite event:

- Composite events
- System events
- Sub-events of other composite events
- Input events, if the composite uses the AND operator.

To remove sub-events from a composite event, use the REMOVE SUBEVENT command.

Options

AND
specifies that the Boolean operator to be associated with this composite's predicate is AND. This means that the composite event will fire when *all* of its sub-events have fired.

Note: The fire status of an empty composite event that uses the AND operator is always FIRED (true).

EVENT(data-value)
specifies the name (1–16 characters) of the composite event being defined. The acceptable characters are A-Z a-z 0-9 $ @ # . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

OR specifies that the Boolean operator to be associated with this composite's predicate is OR. This means that the composite event will fire when *any* of its sub-events fires.

Note: The fire status of an empty composite event that uses the OR operator is always NOTFIRED (false).

SUBEVENTn(data-value)
specifies the name (1–16 characters) of a sub-event to be added to the composite event when the composite is created. The acceptable characters are A-Z a-z 0-9 $ @ # . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

You can specify this option up to 8 times; *n* must be in the range 1–8.

The sub-events that you specify must previously have been defined to the current activity by means of DEFINE INPUT EVENT, DEFINE ACTIVITY, or DEFINE TIMER commands. They must not be sub-events of existing composite events.

Conditions

111 EVENTERR
RESP2 values:

6 The event name specified on the EVENT option is invalid.

7 The event name specified on the EVENT option has already been defined to this activity.

21–28 One or more of the sub-events named on the SUBEVENTn option does not exist. The RESP2 value indicates the first sub-event that does not exist.

16 INVREQ
RESP2 values:

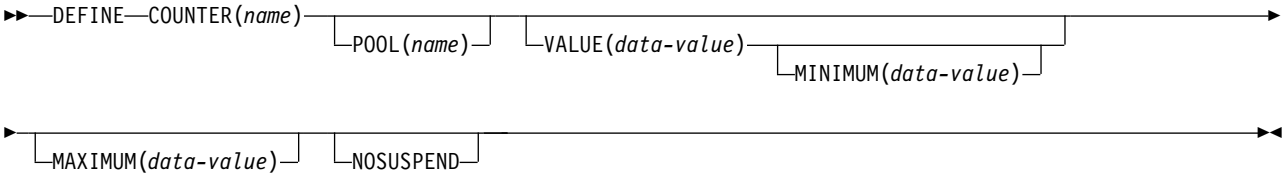
1 The command was issued outside the scope of an activity.

31–38 One or more of the sub-events names specified on the SUBEVENTn option is invalid. The RESP2 value indicates the first invalid sub-event name.

DEFINE COUNTER and DEFINE DCOUNTER

Create a named counter in a named counter pool in the coupling facility. Use COUNTER to create counters that are handled as fullword signed counters and DCOUNTER to create counters that are handled as doubleword unsigned counters.

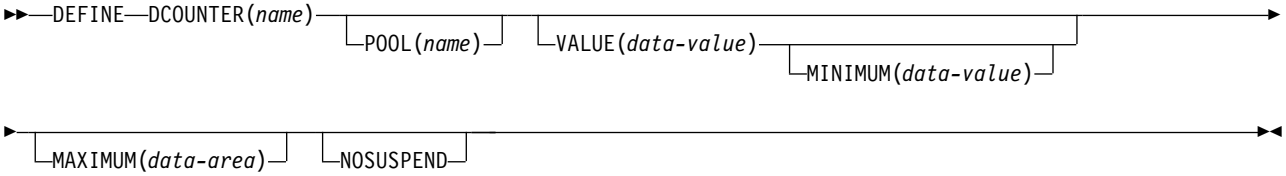
DEFINE COUNTER



Conditions: BUSY, INVREQ

This command is threadsafe.

DEFINE DCOUNTER



Conditions: BUSY, INVREQ

This command is threadsafe.

Description

These counter commands create a new named counter in a named counter pool in the coupling facility.

Although you can use the CICS API to operate with either fullword (signed) or doubleword (unsigned) binary values, the named counter server stores all values as doubleword unsigned values. Overflow conditions might occur if, for example, you define a counter with the DCOUNTER command and try to access it using the COUNTER command. Always access a named counter using commands from the same command set that you used to define the counter.

For information about specifying fullword and doubleword variables on these named counter commands, see “CICS command argument values” on page 3.

Options

COUNTER(*name***)**
Specifies the 16-byte counter name field. The name should start with a letter and may contain letters, digits or underscore characters, padded to 16 bytes with trailing spaces. (Letters must be uppercase. The national use symbols "£", "#" and "@" are allowed wherever letters are allowed). It is recommended that the name should start with a systematic prefix unique to the application. Any counters used by CICS itself will use the prefix "DFH".

DCOUNTER(*name***)**
Specifies the 16-byte counter name field. The name should start with a letter and may contain letters, digits or underscore characters, padded to 16 bytes with trailing spaces. (Letters must be uppercase. The national use symbols "£", "#" and "@" are allowed wherever letters are allowed). It is recommended that the name should start with a systematic prefix unique to the application. Any counters used by CICS itself will use the prefix "DFH".

MAXIMUM(*data-value***)**
Specifies the maximum number for the named counter, using a fullword signed binary value for COUNTER and a doubleword unsigned value for DCOUNTER. This is the maximum number that can be assigned on a GET command, after which the counter must be reset by a REWIND command.

If you omit the MAXIMUM parameter, the named counter is defined with a default maximum of high values (X'7FFFFFFF' for the signed fullword case, or a doubleword filled with X'FF').

MINIMUM(*data-value***)**
Specifies the minimum number for the named counter, using a fullword signed binary value for COUNTER and a doubleword unsigned value for DCOUNTER. This is the value to which a named counter is reset as a result of a REWIND command.

If you specify the MINIMUM parameter, you must also specify a VALUE parameter.

If you omit the MINIMUM parameter, the named counter is defined with a default minimum of low-values (a fullword or doubleword filled with X'00').

NOSUSPEND
Specifies an immediate return to the application program during a coupling facility structure rebuild. If you omit the NOSUSPEND parameter, the request is retried until it is successful. If you specify NOSUSPEND, the request fails immediately, and CICS returns BUSY, RESP2=500.

POOL(*name***)**
Specifies an 8-character string to use as a pool selection parameter to select the pool in which the named counter is to be created. The string can be a logical pool name, or the actual pool name.

Valid characters for the pool selector string are A through Z, 0 through 9, $ @ # and _ (underscore). If *name* is a variable that contains a name that is less than 8 characters, the name must be padded with trailing blanks.

This parameter is optional. If you omit the name of the pool, a pool selector value of 8 blanks is assumed.

If there is no matching entry in the DFHNCOPT options table, CICS uses the name specified on the NCPLDFT system initialization parameter, which specifies the default named counter pool.

For information about generating a named counter options table using the DFHNCO macro, see .

VALUE(*data-value***)**
Specifies the initial number at which the new named counter is to start, using a fullword signed binary value for COUNTER and a doubleword unsigned value for DCOUNTER.

You can specify a number that is equal to, or greater than, the minimum value, up to the maximum value plus 1. If you specify an initial number that is equal to the maximum value plus 1, the counter is created with the counter-at-limit condition set and it cannot be used until it is rewound.

If you omit both the VALUE and MINIMUM parameters, the named counter is created with an initial value of zero. If you omit VALUE but specify a MINIMUM, the translator issues an error; the VALUE parameter is required if you specify the MINIMUM parameter.

Conditions

128 BUSY
RESP2 values:

500 The NOSUSPEND option was specified on the command, and the coupling facility structure is not currently available during a rebuild.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

202 Duplicate counter name. A named counter of this name already exists.

301 The server has reported an error code that is not understood by the named counter interface. Generally, this is not possible unless the interface load module, DFHNCIF, is at a lower maintenance or release level than the server itself.

302 The server cannot create the new named counter because there is not enough space in the named counter pool.

303 An unexpected error, such as structure failure or loss of connectivity, has occurred on a macro used to access the coupling facility. Further information is in message DFHNC0441 in the application job log.

304 The pool selection parameter specified in the program cannot be resolved to a valid server name using the current options table.

305 The interface cannot establish a connection to the server for the selected named counter pool. Further information is in an AXM services message (AXMSC*nnnn*) in the application job log.

306 An abend occurred during server processing of a request. Further information is in a message in the application job log and the server job log.

308 The DFHNCOPT options table module, required to resolve a pool name, cannot be loaded.

309 During processing of the options table, the named counter interface encountered an unknown entry format. Either the options table is not correctly generated, or the DFHNCIF interface load module is not at the same release level as the options table.

310 An options table entry that matches the given pool name specified a user exit program, but the user exit program is not link-edited with the options table and cannot be loaded.

311 A response from the named counter server to the client region interface module, DFHNCIF, indicates that a system-managed rebuild is in progress but the EXEC CICS interface does not recognize the condition. This means that the CICS region is at CICS TS 2.1 or earlier.

403 The POOL parameter contains invalid characters or embedded spaces.

404 The COUNTER parameter contains invalid characters or embedded spaces.

406 The VALUE parameter is invalid. You cannot set the current value to less than the minimum value, or greater than the maximum value plus 1.

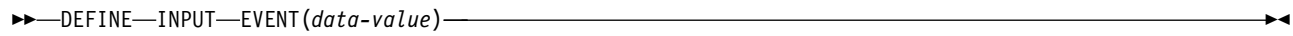
407 The MINIMUM or MAXIMUM parameter is invalid. Either the MAXIMUM parameter specifies a value that is less than the minimum value, or (for COUNTER only) one of the parameters specifies a negative value.

Default action: terminate the task abnormally.

DEFINE INPUT EVENT

Define a BTS input event.

DEFINE EVENT



Conditions: EVENTERR, INVREQ

Description

DEFINE INPUT EVENT defines an input event to BTS. Typically, an input event is passed to an activity by its parent, causing the activity to be activated. (Sometimes, however, the input event originates from outside the process.)

Most events fire on the completion of something, such as an activity or a specified time interval. An input event is different in that it fires after a RUN command that names it is issued.

An activity defines an input event in order to receive notification (via the INPUTEVENT option of the RUN or LINK ACTIVITY commands) of why it has been activated.

Note: System events such as DFHINITIAL are a special type of input event. They are recognized by all activities and do not need to be defined.

Options

EVENT(data-value)
specifies the name (1–16 characters) of the input event being defined. The acceptable characters are A-Z a-z 0-9 $ @ # . - and _. Leading and embedded

blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Conditions

111 EVENTERR
RESP2 values:

6 The event name specified on the EVENT option is invalid.

7 The event name specified on the EVENT option has already been defined to this activity.

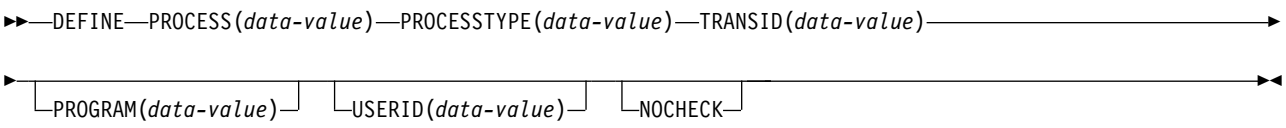
16 INVREQ
RESP2 values:

1 The command was issued outside the scope of an activity.

DEFINE PROCESS

Define a CICS business transaction services process.

DEFINE PROCESS



Conditions: INVREQ, IOERR, NOTAUTH, PROCESSERR, TRANSIDERR

Description

DEFINE PROCESS defines a BTS process. It:

- Adds a new process (for example, a new instance of a business transaction) to the CICS business transaction services system
- Creates the process's root activity.

The name of the program used in the execution of the new process is taken either from the PROGRAM option, or, if PROGRAM is not specified, from the transaction definition pointed to by the TRANSID option.

The transaction attributes specified on the TRANSID and USERID options take effect when the process is activated by a RUN command, but *not* if it is activated by a LINK command—see “RUN” on page 475.

BTS does not commit the addition of the process until the requesting transaction has taken a successful syncpoint.

Options

NOCHECK
specifes that no record is to be written to the repository data set to reserve the name of the process.

Note that the process name must be unique in the repository—see the PROCESS and PROCESSTYPE options—and that BTS does not commit the addition of the process until the requesting transaction has taken a successful syncpoint.

You can use this option to improve BTS performance by removing the write to the repository and its associated logging. However, if you do so be aware that the error of specifying a non-unique process name no longer causes a PROCESSERR condition to be returned on the DEFINE PROCESS command. The error may not be discovered until much later—when syncpoint occurs—making it much harder to debug.

PROCESS(data-value)
specifies a name (1–36 characters) to identify the new process (business transaction instance). The name must be unique within the BTS repository data set on which details of the process are to be stored—see the PROCESSTYPE option. For example, it is valid to issue a DEFINE command on which the PROCESS option specifies a name that is currently in use by another process, *provided that* the PROCESSTYPE option maps to a different underlying repository data set from that on which the first process is defined.

The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = ¬ , ; < > . - and _. Leading and embedded blank characters are also permitted.

If the name is specified as a literal string that is less than 36 characters long, it is padded with trailing blanks up to 36 characters. If the name is specified as a variable whose value is less than 36 characters long, no padding occurs.

PROCESSTYPE(data-value)
specifies the type (1–8 characters) of the new process.

Each process-type maps to a VSAM data set (the repository), on which information about processes of the named type is stored. That is, information about the state of a process (and of its constituent activities) is stored on the repository associated with the process-type to which it belongs. Records for multiple process-types can be stored on the same repository data set.

You can categorize your processes by assigning them to different process-types.

PROGRAM(data-value)
specifies the name (1–8 characters) of the program for the process being added. If no program is specified, the name is taken from the TRANSID definition.

TRANSID(data-value)
specifies the name (1–4 characters) of the transaction under which the process is to run when it is activated by a RUN command.

Note: If the process is activated by a LINK command, it is run under the TRANSID of the transaction that issues the LINK.

The transaction must be defined in the CICS region in which the DEFINE PROCESS command is executed.

USERID(data-value)
specifies the userid (1–8 characters) under whose authority the process is to run when it is activated by a RUN command.

Note: If the process is activated by a LINK command, it is run under the userid of the transaction that issues the LINK.

The value of this field is known as the *defined userid*.

If you omit USERID, the defined userid defaults to the userid under which the transaction that issues the DEFINE command is running—we can call this the *command userid*.

If USERID is specified, CICS performs (at define time) a surrogate security check to verify that the command userid is authorized to use the defined userid. Thus, if you specify USERID, you must authorize the command userid as a surrogate user of the defined userid.

Conditions

16 INVREQ
RESP2 values:

12 The installed PROCESSTYPE is not enabled.

22 The unit of work that issued the DEFINE PROCESS command has already acquired an activity.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access the file associated with the BTS repository data set on which details of the process are to be stored.

102 The user associated with the issuing task is not authorized as a surrogate of the defined userid specified on the USERID option.

108 PROCESSERR
RESP2 values:

2 The process name specified on the PROCESS option is already in use on the BTS repository data set associated with the PROCESSTYPE option.

9 The process-type specified on the PROCESSTYPE option could not be found.

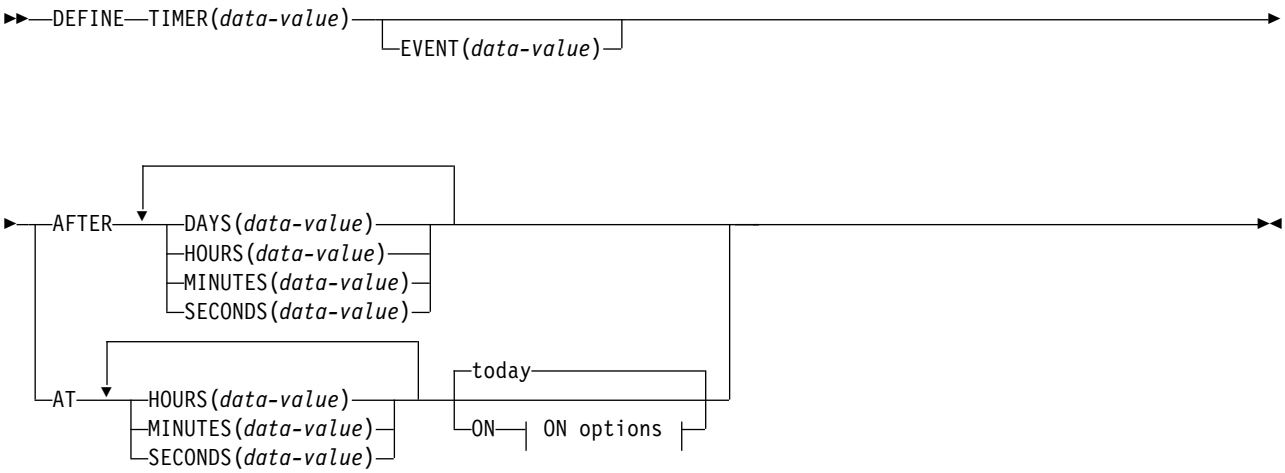
16 The process name specified on the PROCESS option contains an invalid character or characters.

28 TRANSIDERR
The transaction identifier specified on the TRANSID option is not defined to CICS.

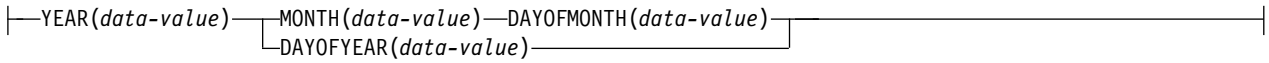
DEFINE TIMER

Define a BTS timer.

DEFINE TIMER



ON options:



Conditions: EVENTERR, INVREQ, TIMERERR

Description

DEFINE TIMER defines a BTS timer which will expire after a specified interval, or at a specified time and date. When a timer is defined, an associated event is also defined, in the event pool of the current activity. The name of the associated event defaults to the name of the timer. When the timer expires, its associated event fires.

Note:

1. All dates and times refer to local time.
2. A timer that specifies a time and date that has already passed expires immediately. Similarly, if the requested interval is zero, the timer expires immediately.

Options

AFTER
specifies the interval of time that is to elapse before the timer is to expire.

You must specify one or more of DAYS(0–999), HOURS(0–23), MINUTES(0–59), and SECONDS(0–59). For example, HOURS(1) SECONDS(3) means one hour and three seconds (the minutes default to zero).

AT specifies the time at which the timer is to expire.

You must specify one or more of HOURS(0–23), MINUTES(0–59), and SECONDS(0–59). For example:

- HOURS(1) means 1 a.m.
- HOURS(15) MINUTES(15) means 3:15 p.m.
- MINUTES(15) means 0:15 a.m.

DAYOFMONTH(*data-value***)**
specifies, as a fullword binary value in the range 1–31, the day-of-the-month on which the timer is to expire.

DAYOFYEAR(*data-value***)**
specifies, as a fullword binary value in the range 1–366, the day-of-the-year on which the timer is to expire. For example, DAYOFYEAR(1) specifies 1st January.

DAYS(*data-value***)**
specifies a fullword binary value in the range 0–999. This is a suboption of the AFTER option. For its use and meaning, see AFTER.

The default value is zero.

EVENT(data-value)
specifies the name (1–16 characters) of the event to be associated with the timer. The acceptable characters are A-Z a-z 0-9 $ @ # . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

The default event name is the name of the timer.

HOURS(*data-value***)**
specifies a fullword binary value in the range 0–23. This is a suboption of the AFTER and AT options. For its use and meaning, see these options.

The default value is zero.

MINUTES(*data-value***)**
specifies a fullword binary value in the range 0–59. This is a suboption of the AFTER and AT options. For its use and meaning, see these options.

The default value is zero.

MONTH(*data-value***)**
specifies, as a fullword binary value in the range 1–12, the month in which the timer is to expire.

ON specifies the date at which the timer is to expire, as a combination of the YEAR, MONTH, DAYOFMONTH, and DAYOFYEAR options.

If the ON option is not specified, the default date is today.

SECONDS(*data-value***)**
specifies a fullword binary value in the range 0–59. This is a suboption of the AFTER and AT options. For its use and meaning, see these options.

The default value is zero.

TIMER(*data-value***)**
specifies the name (1–16 characters) of the timer. The acceptable characters are A-Z a-z 0-9 $ @ # . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

YEAR(*data-value***)**
specifies, as a fullword binary value in the range 0–2040, the year in which the timer is to expire.

Conditions

111 EVENTERR
RESP2 values:

6 The event name specified on the EVENT option is invalid.

7 The event name specified on the EVENT option (or the default event name taken from the timer name) has already been defined to this activity.

16 INVREQ
RESP2 values:

1 The command was issued outside the scope of a currently-active activity.

11 An invalid interval was specified.

12 An invalid date or time was specified.

115 TIMERERR
RESP2 values:

14 The timer name specified on the TIMER option is invalid.

15 The timer name specified on the TIMER option has already been defined to this activity.

Examples

```
DEFINE TIMER() AT HOURS(15)
```

defines a timer that will expire at 3 p.m. today (or immediately if the local time is already later than 3 p.m.).

```
DEFINE TIMER() AT HOURS(15) ON YEAR(2001) MONTH(11) DAYOFMONTH(3)
```

defines a timer that will expire at 3 p.m. on 3rd November 2001.

```
DEFINE TIMER() AT HOURS(15) ON YEAR(2001) DAYOFYEAR(32)
```

defines a timer that will expire at 3 p.m. on 1st February 2001.

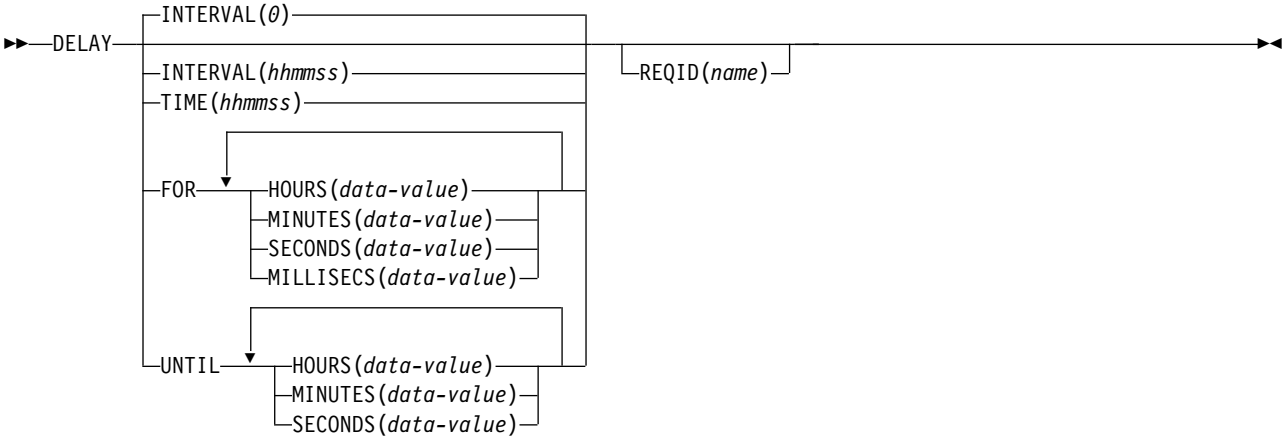
```
DEFINE TIMER() AT HOURS(8) ON YEAR(1997) MONTH(1) DAYOFMONTH(1)
```

defines a timer that expires immediately.

DELAY

Delay the processing of a task.

DELAY



Conditions: EXPIRED, INVREQ

This command is threadsafe only when the interval is 0.

Note for dynamic transaction routing: Using DELAY with REQID if later CANCELed could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. See Affinity for more information about transaction affinities.

Description

DELAY suspends the processing of the issuing task for a specified interval of time or until a specified time of day. It supersedes any previously initiated POST command for the task.

It is possible to specify intervals in milliseconds (ms) however CICS checks for delay expiry every 250 ms so the actual interval might vary depending on where in the scan cycle your request is made.

The default is INTERVAL(0), but for C the default is FOR HOURS(0) MINUTES(0) SECONDS(0).

Options

FOR
specifies the duration of the delay.

HOURS(*data-value***)**
a fullword binary value in the range 0–99.

INTERVAL(*hhmmss***)**
specifies, in packed decimal format, the interval of time that is to elapse from the time when the DELAY command is issued. The **mm** and **ss** are in the range 0–59. The time specified is added to the current clock time by CICS when the command is executed to calculate the expiration time.

When using the C language, you are recommended to use the FOR/UNTIL HOURS, MINUTES, and SECONDS options as C does not provide a packed decimal data type. You may use INTERVAL, but if the value specified is **not** an

integer constant, the application is responsible for ensuring that the value passed to CICS is in packed decimal format.

MILLISECS(*data-value***)**
specifies a fullword binary value in the range 0–999, when HOURS, MINUTES or SECONDS are also specified, or 0-359999999 when MILLISECS is the only option specified.

MINUTES(*data-value***)**
specifies a fullword binary value in the range 0–59, when HOURS, SECONDS or MILLISECS are also specified, or 0–5999 when MINUTES is the only option specified.

REQID(*name***)**
specifies a name (1–8 characters), which should be unique, to identify the DELAY request. Using this option to specify an application-defined name enables another transaction to cancel the DELAY request.

To enable other tasks to cancel unexpired DELAY requests, you must make the request identifier dynamically available. For example, storing it in a TS queue, whose name is known to other applications that may want to cancel the DELAY request, is one way you can pass a request identifier to other transactions.

SECONDS(*data-value***)**
specifies a fullword binary value in the range 0–59, when HOURS, MINUTES or MILLISECS are also specified, or 0–359 999 when SECONDS is the only option specified.

TIME(*hhmmss***)**
specifies, in packed decimal format, the time when the task should resume processing.

When using the C language, you are recommended to use the FOR/UNTIL HOURS, MINUTES, and SECONDS options as C does not provide a packed decimal data type. You may use TIME, but if the value specified is **not** an integer constant, the application is responsible for ensuring that the value passed to CICS is in packed decimal format. See the section about expiration times in Interval control.

UNTIL
specifies the time at the end of the delay and when the task should resume processing.

Conditions

31 EXPIRED
occurs if the time specified has already expired when the command is issued. If you specify very short intervals of less than 250 ms it is likely the time will have lapsed before the command is issued. You will need to use exception handling if short delay intervals are requested.

Default action: ignore the condition.

16 INVREQ
RESP2 values:

4 Hours are out of range.

5 Minutes are out of range.

6 Seconds are out of range.

22 Milliseconds are out of range.

also occurs (RESP2 not set) if the DELAY command is not valid for processing by CICS.

Default action: terminate the task abnormally.

Examples

The following example shows you how to suspend the processing of a task for five minutes:

```
EXEC CICS DELAY
     INTERVAL(500)
     REQID('GXLBZQMR')
```

```
EXEC CICS DELAY FOR MINUTES(5)
```

The following example shows you how, at 09:00, to suspend the processing of a task until 12:45:

```
EXEC CICS DELAY
     TIME(124500)
     REQID('UNIQCODE')
```

There are two ways to enter the time under UNTIL.

- A combination of at least two of HOURS(0–99), MINUTES(0–59) and SECONDS(0–59). HOURS(1) SECONDS(3) would mean one hour and three seconds (the minutes default to zero).
- Any one of HOURS(0–99), MINUTES(0–5999), or SECONDS(0–359 999). HOURS(1) means one hour. MINUTES(62) means one hour and two minutes. SECONDS(3723) means one hour, two minutes, and three seconds.

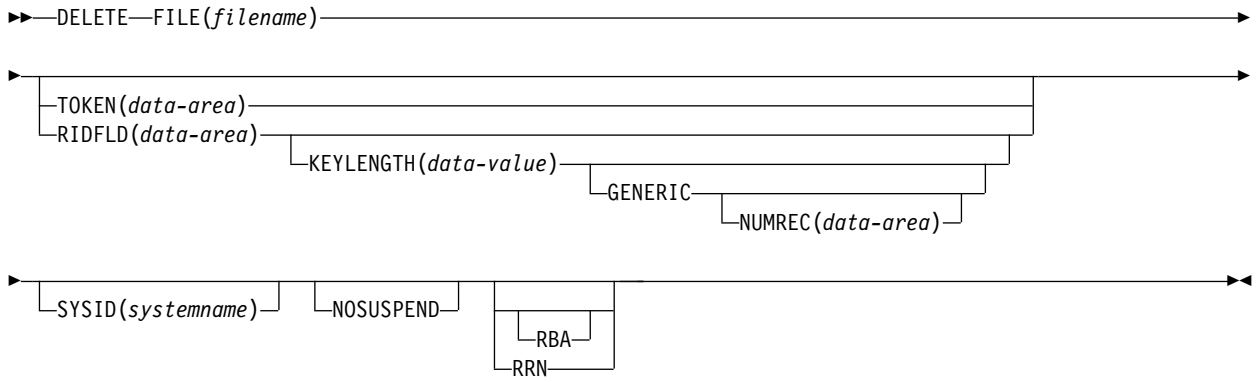
Under FOR you can enter the time in both ways that apply to UNTIL and additionally you can specify a delay to include milliseconds or specify the delay completely in milliseconds. To specify fractions of a second on a delay, code the MILLISECS parameter in the range 0-999 in addition to other time units. To specify the delay purely in milliseconds, code the MILLISECS parameter in the range 0-359999999. The following example shows you how to suspend the processing of a task for 15000 milliseconds:

```
EXEC CICS DELAY
     FOR MILLISECS(15000)
     REQID('UNIQCODE')
```

DELETE

Delete a record from a file - VSAM KSDS, VSAM RRDS, and data tables only.

DELETE



Conditions: CHANGED, DISABLED, DUPKEY, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, LOADING, LOCKED, NOTAUTH, NOTFND, NOTOPEN, RECORDBUSY, SYSIDERR

This command is threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over an IPIC connection to a remote CICS region.
- Defined as either local VSAM or RLS.

This command is not threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over a non-IPIC connection.
- Defined as a shared data table, coupling facility data table, or BDAM file.

Description

The **DELETE** command deletes a record from a file on a KSDS, a path over a KSDS, a CICS or user-maintained data table, or an RRDS. You cannot delete from a VSAM ESDS or a BDAM file. **All references to KSDS apply equally to CICS maintained data tables and, except where stated otherwise, to paths over a KSDS.** The file can be on a local or a remote system. You identify, in the RIDFLD option, the specific record to be deleted.

You can delete a group of records with a single invocation of this command, identifying the group by the GENERIC option (not available for RRDS). The **DELETE** command with GENERIC option should be used with caution. If the generic key specified will match a very large number of records, then the CICS unit of work will be holding a very large number of record locks until syncpoint. This could cause storage issues within the CICS region, or if the file is an RLS file, it could cause excessive use of coupling facility resources and affect other systems. Instead, consider using a series of generic deletes (by using generic keys that will match less records) and syncpoint after each range of records has been deleted.

You can also use this command to delete a single record that has previously been retrieved for update (by a READ UPDATE command). In this case, you must not specify the RIDFLD option.

When this command is used to delete records from a CICS-maintained data table, the update is made to both the source VSAM KSDS data set and the in-memory data table.

When this command is used to delete records from a user-maintained data table, the update is made only to the in-memory data table.

When this command is used to delete records from a coupling facility data table, the update is made only to the data table in the coupling facility.

Options

FILE(*filename***)**
Specifies the name of the file to be accessed.

If SYSID is specified, the data set to which this file refers is assumed to be on a remote system irrespective of whether the name is defined to CICS. Otherwise, the resource definition is used to find out whether the data set is on a local or a remote system.

GENERIC (*VSAM KSDS only***)**
Specifies that the search key is a generic key with a length specified in the KEYLENGTH option. The search for a record is satisfied when a record is found with a key that has the same starting characters (generic key) as those specified.

KEYLENGTH(*data-value***)**
Specifies the length (halfword binary) of the key that has been specified in the RIDFLD option, except when RBA or RRN is specified, in which case it is not valid. This option must be specified if GENERIC is specified, and it can be specified whenever a key is specified. However, if the length specified is different from the length defined for the data set and the operation is not generic, the INVREQ condition occurs.

The INVREQ condition also occurs if you specify GENERIC, and the KEYLENGTH is not less than the length specified in the VSAM definition.

Do not specify a zero value for KEYLENGTH because the results are unpredictable.

For remote files, you can specify the KEYLENGTH in the FILE definition. If KEYLENGTH is not defined there, and is not specified in the application program, and the key is longer than 4 characters, the default value is 4.

NOSUSPEND (RLS only)
Specifies that the request is not to wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK.

Note: Requests that specify NOSUSPEND wait for at least 1 second before CICS returns the RECORDBUSY response.

NUMREC(*data-area***) (VSAM KSDS only)**
Specifies a halfword binary data area that CICS sets to the number of deleted records.

RBA
(VSAM KSDS base data sets only, not paths) Specifies that the record identification field specified in the RIDFLD option contains a relative byte address. Use this option only when deleting records using relative byte addresses instead of keys to identify the records.

You cannot use RBA for:

- User-maintained data tables
- Coupling facility data tables
- Any files opened in RLS access mode

- KSDS files that can hold more than 4 GB of data

RIDFLD(*data-area***)**
Specifies the record identification field. The contents can be a key, a relative byte address (RBA), or a relative record number. For a relative byte address or a relative record number, the format of this field must be fullword binary. For a relative byte address, the RIDFLD can be greater than or equal to zero. For a relative record number, the RIDFLD can be greater than or equal to 1.

The contents must be a key for user-maintained data tables or coupling facility data tables.

You must specify this option if you have also specified GENERIC.

RRN (*VSAM RRDS only***)**
Specifies that the record identification field specified in the RIDFLD option contains a relative record number. Use this option only with files referencing relative record data sets.

SYSID(*systemname***)**
Specifies the name (1-4 characters) of the system the request is directed to.

If you specify SYSID, and omit both RBA and RRN, you must also specify KEYLENGTH; it cannot be found in the resource definition.

TOKEN(*data-area***)**
Specifies, as a fullword binary value, a unique identifier for this DELETE request. Use this identifier to associate the delete request with a record returned on a previous READ UPDATE or BROWSE for UPDATE request. The value to use is the value returned in the TOKEN held by the earlier READ UPDATE or BROWSE for UPDATE request.

TOKEN can be function shipped. However, if a request specifying TOKEN is function shipped to a member of the CICS Family of products that does not recognize this option, the request fails.

Conditions

105 CHANGED
RESP2 values:

109 A **DELETE** command (without RIDFLD) is issued for a file that is a defined as a coupling facility data table using the contention update model and the record has been changed since the application program read it for update. To perform the **DELETE** successfully, repeat the read for update to get the latest version of the record, and try the **DELETE** command again.

Default action: terminate the task abnormally.

84 DISABLED
RESP2 values:

50 A file is disabled. A file can be disabled because:
- It was initially defined as disabled and has not since been enabled.
- It has been disabled by a **SET FILE** or a **CEMT SET FILE** command.

This condition cannot occur when the **DELETE** command follows any read with the UPDATE option.

Default action: terminate the task abnormally.

15 DUPKEY

RESP2 values:

140 A record is accessed by way of an alternate index with the NONUNIQUEKEY attribute, and another alternate index record with the same key follows.

Default action: terminate the task abnormally.

12 FILENOTFOUND

RESP2 values:

1 The file name referred to in the FILE option cannot be found in the file resource definition.

Default action: terminate the task abnormally.

21 ILLOGIC

RESP2 values:

110 A VSAM error occurs that is not in one of the other CICS response categories.

See EIBRCODE in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

16 INVREQ

RESP2 values:

20 Delete operations are not allowed according to the resource definition.

21 A DELETE command is issued for a file referring to a VSAM ESDS.

22 A generic delete is issued for a file that is not a VSAM KSDS.

25 The KEYLENGTH and GENERIC options are specified, and the length specified in the KEYLENGTH option is greater than or equal to the length of a full key.

26 The KEYLENGTH option is specified (but the GENERIC option is not specified), and the specified length does not equal the length defined for the data set to which this file refers.

27 A **DELETE** command is issued for a file referring to a BDAM data set.

31 A **DELETE** command without the RIDFLD option is issued for a file for which no previous **READ UPDATE** command has been issued.

42 The KEYLENGTH and GENERIC options are specified, and the length specified in the KEYLENGTH option is less than zero.

44 The **DELETE** command does not conform to the correct format for a user-maintained or coupling facility data table; for example if RBA was specified.

47 A DELETE instruction includes a token whose value cannot be matched against any token in use for an existing read for UPDATE request.

51 A DELETE command specifying the RBA or XRBA keyword is issued against a KSDS file that is being accessed in RLS mode. RLS does not support relative byte address (RBA) access to KSDS files.

55 NOSUSPEND is specified for a non-RLS file.

56 An attempt to update a recoverable coupling facility data table has

failed because the current unit of work has already updated 1024 recoverable coupling facility data tables. You cannot update more than 1024 recoverable coupling facility data tables within a unit of work

57 DELETE operations are not allowed because the associated VSAM data set has an AVAILABILITY state of RREPL, which disallows such operations except those from replication programs.

59 XRBA was specified, but the data set is not an extended ESDS.

Default action: terminate the task abnormally.

17 IOERR

RESP2 values:

120 There is an I/O error during the file control operation. An I/O error is any unusual event that is not covered by a CICS condition.

For VSAM files, IOERR usually indicates a hardware error. Further information is available in the EXEC interface block; for details, see EIB fields.

For a coupling facility data table, an IOERR indicates a bad response returned from a coupling facility access.

Default action: terminate the task abnormally.

54 ISCINVREQ

RESP2 values:

70 The remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

94 LOADING

RESP2 values:

104 A delete request is issued for a user-maintained data table that is currently being loaded. A user-maintained data table cannot be modified during loading.

LOADING is also returned for a coupling facility data table if the delete request is for a key that is not yet loaded. A coupling facility data table can be modified during loading, but only if the requested key is within the range of those records already loaded.

The LOADING response can also be returned for a coupling facility data table that has failed during loading. For more information about what happens if the load for a coupling facility data table fails, see the description of the XDTLC global user exit in the Data tables management exits (XDTRD, XDTAD, and XDTLC).

If your application programs encounter the LOADING condition persistently or too frequently, check that this condition is not caused by conflicting file definitions that reference the same data set.

Default action: terminate the task abnormally.

100 LOCKED

RESP2 values:

106 An attempt is made to delete a record specifying the RIDFLD, but a *retained* lock exists against this key (see "Retained and active locks" on page 127). If the request specifies the GENERIC keyword, all possible

records are deleted, but the locked records remain. The number of records deleted is returned by NUMREC.

The LOCKED condition can also occur for a DELETE request to a recoverable CFDT that uses the locking model, if the record being read is locked by a retained lock. See the Coupling facility data table retained locks for information about investigating retained locks on records in a coupling facility data table.

Default action: abend the task with code AEX8.

70 NOTAUTH

RESP2 values:

101 A resource security check has failed on FILE(filename).

Default action: terminate the task abnormally.

13 NOTFND

RESP2 values:

80 An attempt to delete a record based on the search argument provided is unsuccessful.

For user-maintained data and coupling facility data tables, this condition occurs if an attempt to delete a record is unsuccessful because there is no entry with the specified key in the data table. This can occur on an attempt to delete a record using a DELETE without RIDFLD, if the delete is associated with a READ UPDATE request for a record that this transaction has deleted (using DELETE with RIDFLD) after it was read for update.

This does not mean that there is no such record in the source data set (if the table was created from one); it might be that such a record is present but was either rejected during initial loading by the user exit XDTRD, or was later deleted from the data table.

For coupling facility data tables, this condition can also occur when a DELETE command (without a RIDFLD) is issued to a coupling facility data table using the contention model, and the record has been deleted since it was read for update.

Default action: terminate the task abnormally.

19 NOTOPEN

RESP2 values:

60 NOTOPEN (RESP2 60) is returned for one of the following reasons:

- The requested file is CLOSED and UNENABLED. The CLOSED, UNENABLED state is reached after a CLOSE request has been received against an OPEN ENABLED file and the file is no longer in use. You can also make CLOSED, UNENABLED the initial state, by specifying STATUS(UNENABLED) and OPENTIME(FIRSTREF) on the FILE resource definition.
- The requested file is OPEN and in use by other transactions, but a CLOSE request against the file has been received.
- A DELETE command is issued against a data set that is quiesced, or is being quiesced, as a result of a SET DSNAME QUIESCED or IMMQUIESCED command.
- The requested file is CLOSED and ENABLED, so CICS has tried to open the file as part of executing the request. This file open has

failed for some reason. Examine the console for messages that explain why the file open has been unsuccessful.

This condition does not occur if the request is made to a CLOSED, DISABLED file. In this case, the DISABLED condition occurs.

This condition also cannot occur when deleting a record just read for update.

Default action: terminate the task abnormally.

101 RECORDBUSY
RESP2 values:

107 The NOSUSPEND keyword is specified for the deletion of a record that is locked by a VSAM *active* lock (see "Retained and active locks").

If the request specifies the GENERIC keyword, all possible records are deleted except for the locked records which remain. The number of records deleted is returned by NUMREC.

Default action: abend the task with code AEX9.

53 SYSIDERR
RESP2 values:

130 The SYSID option specifies a name that is not the local system or a remote system (made known to CICS by defining a CONNECTION). SYSIDERR also occurs when the link to the remote system is closed.

131 For a coupling facility data table, the connection to the coupling facility data table server has failed. This failure could occur because the server itself has failed, or the server is available, but CICS has failed to connect to it.

132 The **DELETE** command is issued against a coupling facility data table that no longer exists, probably because of a coupling facility failure, in which case the coupling facility data table server also fails.

Default action: terminate the task abnormally.

Retained and active locks

RECORDBUSY refers to active locks and LOCKED refers to retained locks:

- DELETE requests for records that have retained locks are always rejected with a LOCKED response.
- DELETE requests for records that have active locks wait for the lock to be released, except when the NOSUSPEND keyword is specified, in which case CICS returns the RECORDBUSY response.

Examples

The following example shows you how to delete a group of records in a VSAM data set:

```
EXEC CICS DELETE
     FILE('MASTVSAM')
     RIDFLD(ACCTNO)
     KEYLENGTH(len)
     GENERIC
     NUMREC(NUMDEL)
```

DELETE ACTIVITY

Delete a BTS child activity.

DELETE



Conditions: ACTIVITYBUSY, ACTIVITYERR, INVREQ, IOERR, LOCKED

Description

DELETE ACTIVITY removes a child activity from the BTS repository data set on which it is defined. The child activity's completion event is removed from the parent's event pool. Any descendants of the child activity are also deleted.

The activity to be deleted must be a child of the activity that issues the DELETE command. To be eligible for deletion, the child activity must be in one of the following processing states (modes):

- COMPLETE—completed normally, abnormally, or previously canceled.
- INITIAL—not yet run, or reset by means of a RESET ACTIVITY command.

For a description of all possible processing states, see Processing modes .

Note: A child activity that is not deleted explicitly by means of a DELETE ACTIVITY command is deleted automatically by CICS when its parent completes.

Options

ACTIVITY(data-value)
specifies the name (1–16 characters) of the child activity to be deleted.

Conditions

107 ACTIVITYBUSY
RESP2 values:

19 The request timed out. It may be that another task using this activity-record has been prevented from ending.

109 ACTIVITYERR
RESP2 values:

8 The activity named on the ACTIVITY option could not be found.

14 The child activity named on the ACTIVITY option is not in COMPLETE or INITIAL mode, and is therefore ineligible for deletion.

16 INVREQ
RESP2 values:

4 The DELETE ACTIVITY command was issued outside the scope of a currently-active activity.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

DELETE CHANNEL

Delete a named channel and all the containers that are in it.

DELETE CHANNEL



Conditions: CHANNELERR

This command is threadsafe.

Description

DELETE CHANNEL deletes the specified channel and all the containers that are in it. When you delete a channel and its containers:

- Any data that is in the containers is discarded.
- All storage that relates to the channel and to its containers is released.

The application program that issues the **DELETE CHANNEL** command must be the program that owns the channel. The program that owns the channel is the program that created the channel by naming it on one of the following commands:

- **LINK PROGRAM CHANNEL**
- **MOVE CONTAINER CHANNEL TOCHANNEL**
- **PUT CONTAINER CHANNEL**
- **PUT64 CONTAINER**
- **RETURN TRANSID CHANNEL**
- **START TRANSID CHANNEL**
- **XCTL PROGRAM CHANNEL**
- **WEB RECEIVE TOCHANNEL**
- **WEB CONVERSE TOCHANNEL**

An application program cannot delete the following channels:

- The current channel for the application program, that is, the channel with which the program was invoked.
- Any channel that the application program did not create.
- Any channel that is read-only.
- The transaction channel DFHTRANSACTION.

Options

CHANNEL(data-value)
Specifies the 1–16 character name of the channel that is to be deleted. Every container that is owned by the channel is deleted, and the channel itself is deleted.

Conditions

122 CHANNELERR
RESP2 values:

2 The channel specified on the CHANNEL option could not be found.

3 The channel specified on the CHANNEL option is a read-only channel.

4 The channel specified on the CHANNEL option is the current channel.

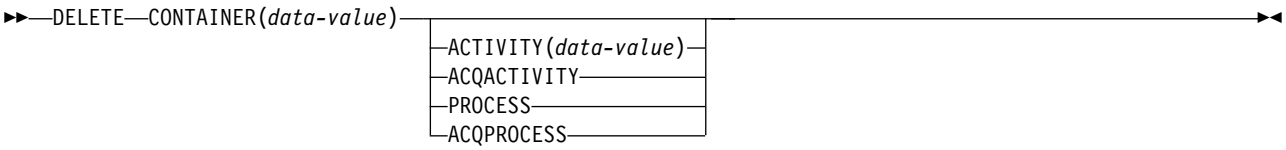
5 The channel specified on the CHANNEL option is the transaction channel.

6 The channel specified on the CHANNEL option is not owned by the calling program.

DELETE CONTAINER (BTS)

Delete a named BTS data-container.

DELETE CONTAINER (BTS)



Conditions: ACTIVITYERR, CONTAINERERR, INVREQ, IOERR, LOCKED, PROCESSBUSY

Description

DELETE CONTAINER (BTS) deletes a BTS data-container and discards any data that it contains.

The container is identified by name and by the process or activity for which it is a container—the process or activity that "owns" it. The activity that owns the container can be identified:

- Explicitly, by specifying one of the PROCESS- or ACTIVITY-related options.
- Implicitly, by omitting the PROCESS- and ACTIVITY-related options. If these are omitted, the current activity is implied.

Note: Process containers can be deleted only by the root activity or by a program that has acquired the process.

Options

ACQACTIVITY
specifies either of the following:

- If the program that issues the command has acquired a process, that the container is owned by the root activity of that process.
- Otherwise, that the container is owned by the activity that the program has acquired by means of an ACQUIRE ACTIVITYID command.

ACQPROCESS
specifies that the container is owned by the process that the program that issues the command has acquired in the current unit of work.

ACTIVITY(data-value)
specifies the name (1–16 characters) of the activity that owns the container. This must be a child of the current activity.

CONTAINER(data-value)
specifies the name (1–16 characters) of the container to be deleted.

PROCESS
specifies that the container to be deleted is owned by the current process—that is, the process that the program that issues the command is executing on behalf of.

Conditions

109 ACTIVITYERR
RESP2 values:

8 The activity named on the ACTIVITY option could not be found.

110 CONTAINERERR
RESP2 values:

10 The container named on the CONTAINER option could not be found.

26 The process container named on the CONTAINER option is read-only. (Process containers can be deleted only by the root activity or by a program that has acquired the process.)

16 INVREQ
RESP2 values:

4 The command was issued outside the scope of a currently-active activity.

15 The ACQPROCESS option was used, but the unit of work that issued the request has not acquired a process.

24 The ACQACTIVITY option was used, but the unit of work that issued the request has not acquired an activity.

25 The PROCESS option was used, but the command was issued outside the scope of a currently-active process.

17 IOERR
RESP2 values:

30 An input/output error has occurred on the repository file.

31 The record on the repository file is in use.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

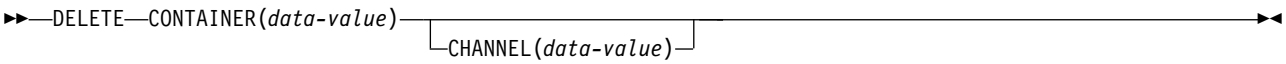
106 PROCESSBUSY
RESP2 values:

13 The request could not be satisfied because the process record is locked by another task.

DELETE CONTAINER (CHANNEL)

Delete a named channel container.

DELETE CONTAINER (CHANNEL)



Conditions: CHANNELERR, CONTAINERERR, INVREQ

This command is threadsafe.

Description

DELETE CONTAINER (CHANNEL) deletes a container from a channel and discards any data that it contains.

The container is identified by name and by the channel for which it is a container - the channel that "owns" it.

The channel that owns the container can be identified explicitly, by specifying the CHANNEL option, or implicitly, by omitting the CHANNEL option. When this option is omitted, the current channel is implied. If there is no current channel, the command is assumed to be an EXEC CICS DELETE CONTAINER (BTS) command. For more information, see "DELETE CONTAINER (BTS)" on page 130.

Options

CHANNEL(data-value)
Specifies the name (1–16 characters) of the channel that owns the container. You can specify the channel name DFHTRANSACTION to use the transaction channel.

CONTAINER(data-value)
Specifies the name (1–16 characters) of the container to be deleted.

Conditions

122 CHANNELERR
RESP2 values:

2 The channel specified on the CHANNEL option could not be found.

3 Either the current channel or the channel specified on the CHANNEL option is read-only.

110 CONTAINERERR
RESP2 values:

10 The container named on the CONTAINER option could not be found.

16 INVREQ
RESP2 values:

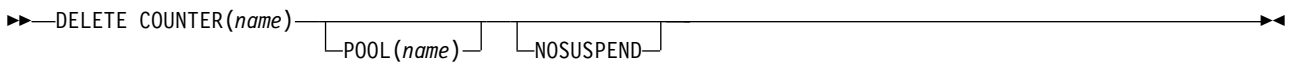
4 The command was issued outside the scope of a currently-active channel.

30 You cannot delete a CICS-defined read-only container.

DELETE COUNTER and DELETE DCOUNTER

Delete the named counter from the specified pool. Use COUNTER for fullword signed counters and DCOUNTER for doubleword unsigned counters.

DELETE COUNTER



Conditions: BUSY, INVREQ

This command is threadsafe.

DELETE DCOUNTER



Conditions: BUSY, INVREQ

This command is threadsafe.

Description

These commands delete the named counter from the specified pool.

Options

COUNTER(*name*)
Specifies the name of the fullword counter to delete. The name can be up to 16 alphanumeric characters. If *name* is a variable that contains a name that is less than 16 characters, the name must be padded with trailing blanks.

DCOUNTER(*name*)
Specifies the name of the doubleword counter to delete. The name can be up to 16 alphanumeric characters. If *name* is a variable that contains a name that is less than 16 characters, the name must be padded with trailing blanks.

NOSUSPEND
Specifies an immediate return to the application program during a coupling facility structure rebuild. If you omit the NOSUSPEND parameter, the request is retried until it is successful. If you specify NOSUSPEND, the request fails immediately, and CICS returns BUSY, RESP2=500.

POOL(*poolname*)
Specifies an 8-character string to use as a pool selection parameter to select the pool in which the named counter resides. The string can be a logical pool name, or the actual pool name.

Valid characters for the pool selector string are A through Z, 0 through 9, $ @ # and _ (underscore). If *name* is a variable that contains a name that is less than 8 characters, the name must be padded with trailing blanks.

This parameter is optional. If you omit the name of the pool, a pool selector value of 8 blanks is assumed.

If there is no matching entry in the DFHNCOPT options table, CICS uses the name specified on the NCPLDFT system initialization parameter, which specifies the default named counter pool.

For information about generating a named counter options table using the DFHNCO macro, see .

Conditions

128 BUSY

RESP2 values:

500 The NOSUSPEND option was specified on the command, and the coupling facility structure is not currently available during a rebuild.

Default action: terminate the task abnormally.

16 INVREQ

RESP2 values:

201 Named counter not found.

301 The server has reported an error code that is not understood by the named counter interface. Generally, this is not possible unless the interface load module, DFHNCIF, is at a lower maintenance or release level than the server itself.

303 An unexpected error, such as structure failure or loss of connectivity, has occurred on a macro used to access the coupling facility. Further information is in message DFHNC0441 in the application job log.

304 The pool selection parameter specified in the program cannot be resolved to a valid server name using the current options table.

305 The interface cannot establish a connection to the server for the selected named counter pool. Further information is in an AXM services message (AXMSC*nnnn*) in the application job log.

306 An abend occurred during server processing of a request. Further information is in a message in the application job log and the server job log.

308 The DFHNCOPT options table module, required to resolve a pool name, cannot be loaded.

309 During processing of the options table, the named counter interface encountered an unknown entry format. Either the options table is not correctly generated, or the DFHNCIF interface load module is not at the same release level as the options table.

310 An options table entry that matches the given pool name specified a user exit program, but the user exit program is not link-edited with the options table and cannot be loaded.

311 A response from the named counter server to the client region interface module, DFHNCIF, indicates that a system-managed rebuild is in progress but the EXEC CICS interface does not recognize the condition. This means that the CICS region is at CICS TS 2.1 or earlier.

403 The POOL parameter contains invalid characters or embedded spaces.

Default action: terminate the task abnormally.

DELETE EVENT

Delete a BTS event.

DELETE EVENT



Conditions: EVENTERR, INVREQ

Description

DELETE EVENT deletes a BTS event that is no longer needed. The event is removed from the current activity's event pool. An event can be deleted whether it has fired or not.

DELETE EVENT can be used to delete only the following types of event:

- Input
- Composite.

DELETE EVENT *cannot* be used to delete:

- Activity completion events. These are implicitly deleted when a response from the completed activity is acknowledged by a CHECK ACTIVITY command issued by the activity's parent; or when a DELETE ACTIVITY command is issued.
- Timer events. These are implicitly deleted when the expiry of the associated timer is acknowledged by a CHECK TIMER command; or when a DELETE TIMER command is issued.
- System events.

Note:

1. If the event to be deleted is included in the predicate of a composite event, it is removed from the predicate's Boolean expression. The fire status of the composite event (FIRED or NOTFIRED) is re-evaluated.
2. Deleting a composite event has no effect on its sub-events.

Options

EVENT(data-value)
specifies the name (1–16 characters) of the event to be deleted.

Conditions

111 EVENTERR
RESP2 values:

4 The event specified on the EVENT option is not recognized by BTS.

16 INVREQ
RESP2 values:

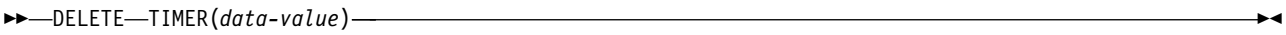
1 The command was issued outside the scope of an activity.

2 The event specified on the EVENT option cannot be deleted because it is a system, timer, or activity completion event.

DELETE TIMER

Delete a BTS timer.

DELETE TIMER



Conditions: INVREQ, TIMERERR

Description

DELETE TIMER deletes a BTS timer. If an event is associated with the timer, the event is also deleted and removed from the current activity's event pool. (There will be no event associated with the timer if the timer has expired and a CHECK TIMER command has been issued.)

The only timers a program can delete are those owned by the current activity. A timer can be deleted whether it has expired or not.

Options

TIMER(data-value)
specifies the name (1–16 characters) of the timer to be deleted.

Conditions

16 INVREQ
RESP2 values:

1 The command was issued outside the scope of a currently-active activity.

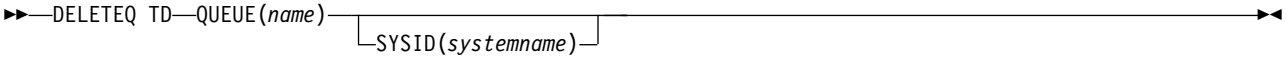
115 TIMERERR
RESP2 values:

13 The timer specified on the TIMER option does not exist.

DELETEQ TD

Delete all transient data.

DELETEQ TD



Conditions: DISABLED, INVREQ, ISCINVREQ, LOCKED, NOTAUTH, QIDERR, SYSIDERR

This command is threadsafe when it is used with a queue in a local CICS region, or function shipped to a remote CICS region over an IPIC connection. It is non-threadsafe when it is function shipped to a remote CICS region over another type of connection.

Description

DELETEQ TD deletes all the transient data associated with a particular intrapartition destination (queue). All storage associated with the destination is released (deallocated). Note that you cannot use this command to delete an *extrapartition* transient data queue. An attempt to do so results in an INVREQ condition.

Options

QUEUE(*name***)**
Specifies the symbolic name (1 - 4 alphanumeric characters) of the queue to be deleted. The named queue must be defined to CICS.

If SYSID is specified, the queue is assumed to be on a remote system, irrespective of how it is defined. Otherwise the resource definition is used to find out whether the queue is on a local or a remote system.

SYSID(*systemname***)**
(remote systems only) Specifies the name (1 - 4 characters) of the system the request is directed.

Conditions

84 DISABLED
Occurs when the queue has been disabled.

Default action: terminate the task abnormally.

16 INVREQ
Occurs if DELETEQ names an extrapartition queue.

Default action: terminate the task abnormally.

54 ISCINVREQ
Occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

100 LOCKED
Occurs when the request cannot be performed because use of the queue has been restricted owing to a unit of work failing indoubt. This can happen on any request for a logically-recoverable queue defined with WAIT(YES) and WAITACTION(REJECT) in the TDQUEUE resource definition.

Specify WAIT(YES) and WAITACTION(QUEUE) in the TDQUEUE resource definition if you want the transaction to wait.

Default action: terminate the task abnormally.

70 NOTAUTH
Occurs when a resource security check has failed on QUEUE(*name*).

Default action: terminate the task abnormally.

44 QIDERR
Occurs if the symbolic destination to be used with DELETEQ TD cannot be found.

Default action: terminate the task abnormally.

53 SYSIDERR
Occurs when the SYSID option specifies a name that is neither the local system

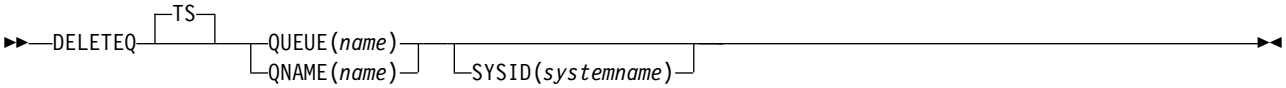
nor a remote system (made known to CICS by defining a CONNECTION or an IPCONN). SYSIDERR also occurs when the link to the remote system is closed.

Default action: terminate the task abnormally.

DELETEQ TS

Delete a temporary storage queue.

DELETEQ TS



Conditions: INVREQ, ISCINVREQ, LOCKED, NOTAUTH, QIDERR, SYSIDERR

This command is threadsafe when it is used with a queue in main storage or auxiliary storage, either in a local CICS region, or function shipped to a remote CICS region over an IPIC connection. The command is non-threadsafe when it is function shipped to a remote CICS region over another type of connection, or when it is used with a queue in a shared temporary storage pool in a z/OS coupling facility that is managed by a temporary storage data sharing server (TS server).

Note for dynamic transaction routing: Using this command might create inter-transaction affinities that adversely affect the use of dynamic transaction routing. For more information about transaction affinities, see Affinity .

Description

DELETEQ TS deletes all the temporary data associated with a temporary storage queue. All storage associated with the queue is freed.

You should delete temporary data as soon as possible to avoid using excessive amounts of storage.

When a recoverable temporary storage queue is deleted, you must issue a sync point before issuing a subsequent **WRITEQ TS** command for the same queue.

Options

QNAME(*name***)**
An alternative to QUEUE, QNAME specifies the symbolic name (1 - 16 characters) of the queue to be deleted. The name must not consist solely of binary zeros and must be unique in the CICS system. If the name has less than 16 characters, you must still use a 16-character field, padded with blanks if necessary.

QUEUE(*name***)**
Specifies the symbolic name (1 - 8 characters) of the queue to be deleted. The name cannot consist solely of binary zeros and must be unique within the CICS system. If the name has less than 8 characters, you must still use an 8-character field, padded with blanks if necessary.

SYSID(*systemname***)**
(Remote and shared queues only) Specifies the system name (1 - 4 characters) identifying the remote system or shared queue pool to which the request is directed. Note that TSMODEL resource definitions do not support specifying a

SYSID for a queue that resides in a temporary storage data sharing pool. Use the QUEUE or QNAME option instead. Using an explicit SYSID for a shared queue pool requires the support of a temporary storage table (TST).

Conditions

16 INVREQ
Occurs in either of the following situations:

- The queue was created by CICS internal code.
- The queue name specified consists solely of binary zeroes.

Default action: terminate the task abnormally.

54 ISCINVREQ
Occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

100 LOCKED
Occurs when the request cannot be performed because use of the queue has been restricted owing to a unit of work failing indoubt.

Default action: terminate the task abnormally.

70 NOTAUTH
Occurs when a resource security check has failed on QUEUE(*name*).

Default action: terminate the task abnormally.

44 QIDERR
Occurs when the specified queue cannot be found in either main or auxiliary storage.

Default action: terminate the task abnormally.

53 SYSIDERR
Occurs in any of the following situations:

- The SYSID option specifies a name that is not the local system or a remote system (made known to CICS by defining a CONNECTION or an IPCONN).
- When IPIC connectivity is used, the local system, the remote system, or both, are not CICS TS 4.2 or later regions.
- The link to the remote system is closed.
- The CICS region in which the temporary storage command is executed fails to connect to the TS server managing the TS pool that supports the referenced temporary storage queue. For example, this situation can occur if the CICS region is not authorized to access the temporary storage server.

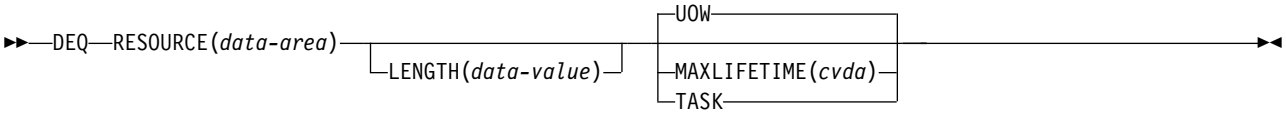
  This condition can also occur if the temporary storage server is not started, or because the server has failed (or been stopped) while CICS continues to run.

Default action: terminate the task abnormally.

DEQ

Schedule use of a resource by a task (dequeue).

DEQ



Conditions: INVREQ, LENGERR

This command is threadsafe if it is defined as LOCAL. It is non-threadsafe if it is defined as GLOBAL.

Note for dynamic transaction routing: Using this command could create inter-transaction affinities that adversely affect the use of dynamic transaction routing, unless the name specified in RESOURCE matches the name specified in an installed ENQMODEL resource definition, that has sysplex-wide scope. See Affinity for more information about transaction affinities.

Description

DEQ causes a resource currently enqueued on by the task to be released for use by other tasks.

If a task enqueues on, but does not dequeue from, a resource, CICS automatically releases the resource during syncpoint processing or when the task is terminated. A resource in the context of this command is any string of 1–255 bytes, established by in-house standards, to protect against conflicting actions between tasks, or to cause single-threading within a program.

When issuing the DEQ command, the resource to be dequeued from must be identified by the method used when enqueuing on the resource. If no enqueue has been issued for the resource, the dequeue is ignored.

If more than one ENQ command is issued for a resource by a task, that resource remains owned by the task until the task issues a matching number of DEQ commands.

When an EXEC CICS DEQ (or ENQ) command is issued for a resource whose name matches that of an installed ENQMODEL resource definition, CICS checks the value of the ENQSCOPE attribute to determine whether the scope is local or sysplex-wide. If the ENQSCOPE is non-blank, CICS treats the ENQ or DEQ as sysplex-wide, and passes a queue name and the resource name to z/OS global resource serialization to manage the enqueue. If the ENQSCOPE attribute is left blank (the default value), CICS treats the DEQ as local to the issuing CICS region. If no ENQMODEL matches the resource name, the scope of the DEQ command is local. For more information on ENQMODEL resources, see ENQMODEL resources.

Options

LENGTH(*data-value***)**
specifies that the resource to be dequeued from has a length given by the data value. The data value is a halfword binary value in the range 1 through 255. If the value you specify is outside this range, a LENGERR condition occurs. If the LENGTH option is specified in an ENQ command, it must also be specified in the DEQ command for that resource, and the values of these options must be the same.

MAXLIFETIME(*cvda***)**
specifies the duration of the ENQ being released. CVDA values are:

UOW The enqueue was acquired with a duration of a unit of work. This is the default value.

Note: For compatibility with previous releases of CICS, a CVDA value of LUW is also supported.

TASK
ENQ was acquired with a duration of a task.

RESOURCE(*data-area***)**
specifies either an area whose address represents the resource to be dequeued from, or a variable that contains the resource (an employee name, for example). In the latter case, you must use the LENGTH option.

Conditions

16 INVREQ
RESP2 values:

2 The MAXLIFETIME option is set with an incorrect CVDA.

Default action: terminate the task abnormally.

22 LENGERR
RESP2 values:

1 The value you specified for the LENGTH option is outside the range 1 through 255.

Default action: terminate the task abnormally.

Examples

The following examples show how to dequeue from a resource:

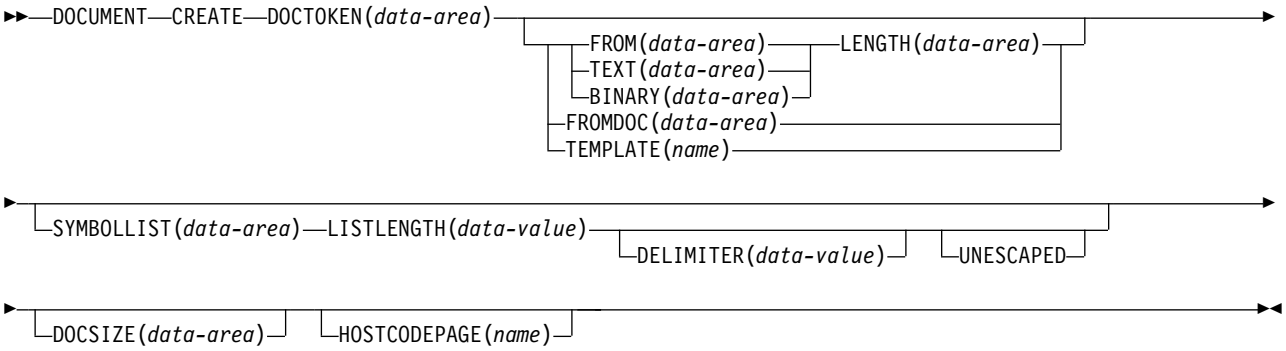
```
EXEC CICS DEQ
     RESOURCE(RESNAME)

EXEC CICS DEQ
     RESOURCE(SOCSECNO)
     LENGTH(9)
```

DOCUMENT CREATE

Create a document.

DOCUMENT CREATE



Conditions: INVREQ, LENGERR, NOTAUTH, NOTFND, SYMBOLERR, TEMPLATERR

This command is threadsafe.

Description

DOCUMENT CREATE signals the start of the document creation process. The document being created can be an empty document, or it can be based on an existing document, a template, or data contained in an application buffer.

Options

BINARY(data-area)
specifies a buffer of data which is to be used as the contents of the new document being created. The data is copied unchanged to the document content and no attempt is made to parse the data for symbol substitution. The purpose of the BINARY option is to allow the application to insert blocks of data that must not undergo conversion to a client code page when the data is sent. Use the LENGTH option to specify the length of this buffer.

DELIMITER(data-value)
specifies an optional 1–byte value used to delimit symbol name-value pairs in the SYMBOLLIST buffer. If this option is not specified, the value defaults to an ampersand. Certain delimiter values (such as the space character) are disallowed, and all of these cause an INVREQ condition on the command if used. The rules are listed in Rules for specifying symbols and symbol lists.

If this option is used, the application must ensure that the DELIMITER does not appear in any symbol value in the SYMBOLLIST buffer. For this reason, the application should not use alphanumeric and other printable characters as the DELIMITER value.

DOCSIZE(data-area)
specifies a binary fullword area that will be updated with the current size of the document in bytes. This is the maximum size of the buffer needed to contain a copy of the document when a RETRIEVE command is issued.

DOCTOKEN(data-area)
specifies a data area to contain the binary token of the document. The area must be 16 bytes in length and will be set to a CICS-generated name by which the document can be referred to in later commands.

FROM(data-area)
specifies that data supplied by the application is to be used to create the contents of the new document. The data content could be a template or a document which was created and retrieved. If the data is a template, symbol substitution will take place where the symbols exist in the symbol table. If the data is a previously retrieved document, the conversion and bookmark tags which were inserted during retrieval will be removed from the content and stored in the internal format required by the API commands. Note that symbol substitution will not be attempted on any unresolved symbols contained in a retrieved document. Use the LENGTH option to specify the length of this buffer.

FROMDOC(data-area)
specifies the binary token (see the **DOCTOKEN** option) of a document whose contents are to be copied to the new document being created. All bookmark and conversion tags are copied to the new document. The symbol table will be not be copied.

HOSTCODEPAGE(name)
specifies the name of the host code page that the data being added is encoded in. This option applies to the TEXT, SYMBOL and TEMPLATE options only. The name must be eight characters long; if it is shorter than eight characters it must be padded on the right with blanks.

The standard CICS form of a host code page name consists of the code page number (or more generally CCSID) written using 3 to 5 decimal digits as necessary then padded with trailing spaces to 8 characters. For code page 37, which is fewer than 3 digits, the standard form is 037. CICS now also accepts any decimal number of up to 8 digits (padded with trailing spaces) in the range 1 to 65535 as a code page name, even if it is not in the standard form.

Note that the HOSTCODEPAGE parameter must specify an EBCDIC-based code page if any symbol processing will be required, as the delimiters used for symbol and symbol list processing are assumed to be in EBCDIC.

LENGTH(data-value)
specifies the length, as a fullword binary value, of the buffer containing the TEXT, BINARY or FROM data.

LISTLENGTH(data-value)
specifies the length, as a fullword binary value, of the symbol list.

SYMBOLLIST(*data-area*)
specifies a buffer which contains a symbol list. Use the LISTLENGTH option to specify the length of this buffer. A symbol list is a character string consisting of one or more symbol definitions separated by ampersands. Each symbol definition consists of a name, an equals sign, and a value. Here is an example of a symbol list:

```
applid=IYCQ&jobname=test
```

By default, symbols in the symbol list are separated by the & character, but you can override this by using the DELIMITER keyword to specify a different symbol separator. Rules for specifying symbols and symbol lists lists the rules which apply when setting symbols using a SYMBOLLIST.

TEMPLATE(name)
specifies the 48-byte name of a template. The template must be defined to CICS using RDO. If the name is shorter than 48 bytes, it must be padded on the right with blanks.

Note: If you insert a template before the symbols contained in it are set, the symbols will never be substituted. This can occur if you create a document from a template without specifying a symbol list.

TEXT(data-area)
specifies a buffer of data which is to be used as the contents of the new document being created. The data is copied unchanged to the document content and no attempt is made to parse the data for symbol substitution. The data will be marked as requiring conversion to the client code page when the document is sent. Use the LENGTH option to specify the length of this buffer.

UNESCAPED
prevents CICS from unescaping symbol values contained in the SYMBOLLIST buffer. If this option is used, plus signs are not converted to spaces, and sequences such %2B are not converted to single byte values.

The UNESCAPED option does not allow you to include the character that you have used as the symbol separator within a symbol value in a symbol list. If you want to use the UNESCAPED option, choose a symbol separator that will never be used within a symbol value.

Conditions

INVREQ
RESP2 value:

1 The retrieved document specified on the FROM option is not in a valid RETRIEVE format.

LENGERR
RESP2 value:

1 The value specified for LENGTH is invalid. The value is negative.

9 The value specified for LISTLENGTH is invalid. Value must be between 1 and (16M - 1).

70 NOTAUTH
The command has failed a resource security check. (If the NOTAUTH condition is not handled, applications that receive it may abend with code AEY7.)

Note that the **EXEC CICS DOCUMENT** commands reference document templates using the 48-character name of the template (as specified in the TEMPLATENAME attribute of the DOCTEMPLATE resource). However, security checking for the commands uses the name of the DOCTEMPLATE resource definition that corresponds to the TEMPLATENAME attribute. If resource security checking is in place, the user ID for the transaction must have READ access to this DOCTEMPLATE resource definition.

RESP2 value:

101 The user ID for the transaction does not have READ access to the DOCTEMPLATE resource definition for the document template named by the TEMPLATE option.

13 NOTFND
RESP2 values:

2 The document specified on the FROMDOC option could not be found or was named incorrectly.

3 The template specified on the TEMPLATE option could not be found or was named incorrectly.

7 The host code page specified on the HOSTCODEPAGE option could not be found or was named incorrectly.

8 The value specified for DELIMITER is not valid.

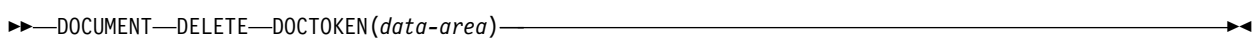
116 SYMBOLERR
A symbol specified in the symbol list does not conform to the naming rules for symbols. RESP2 contains the offset of the symbol in the list.

117 TEMPLATERR
The template specified on the DOCTEMPLATE resource does not exist or an invalid #set, #include or #echo command was encountered while processing the supplied template data. RESP2 contains the offset of the invalid command.

DOCUMENT DELETE

Delete a document.

DOCUMENT DELETE



Conditions: NOTFND

This command is threadsafe.

Description

The **DOCUMENT DELETE** command enables you to delete documents that are no longer required during a transaction. The command allows the application to request deletion of a document and all storage related to the document. On execution of this command, the storage allocated to the document is released immediately. If the **DOCUMENT DELETE** command is not invoked, the document exists until the application ends.

Options

DOCTOKEN(data-area)
specifies the 16-byte binary token of the document to be deleted.

Conditions

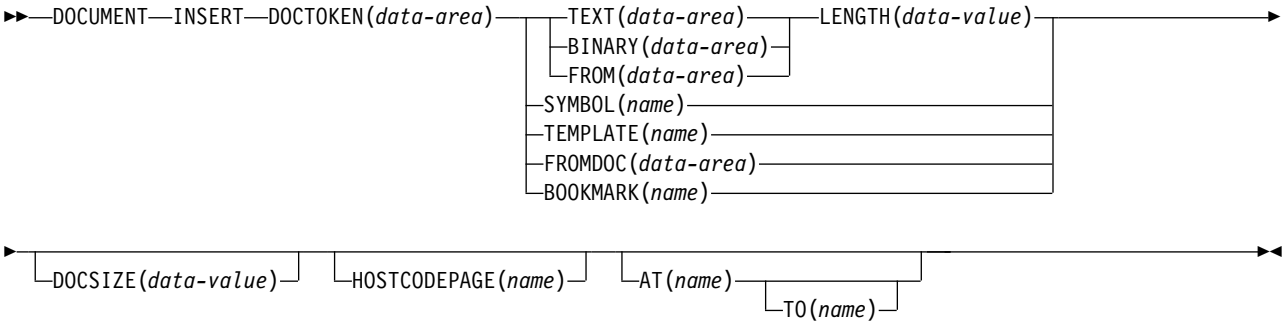
13 NOTFND
RESP2 values:

1 The document has not been created, or the name is incorrectly specified.

DOCUMENT INSERT

Insert document objects.

DOCUMENT INSERT



Conditions: DUPREC, INVREQ, LENGERR, NOTAUTH, NOTFND, TEMPLATERR

This command is threadsafe.

Description

DOCUMENT INSERT allows the application to insert document objects at insertion points in the document. The insertion points (bookmarks) define relative positions in the document. Bookmarks must be defined before being referenced. Data is always inserted after the position identified by the bookmark.

Options

AT(*name***)**
specifies the 16-byte symbolic name of a bookmark which identifies the position of the insertion point in the document. Data is inserted after the bookmark, and any data following the bookmark is shifted down. The application can use a combination of the AT and TO options to perform an overlay operation. If the AT operand is not specified, the data is inserted at the end of the document. A pre-defined bookmark of TOP is provided to allow the application to insert data at the beginning of the document.

BINARY(*data-area***)**
specifies a buffer of data to be inserted into the document. The data is copied unchanged to the insertion point in the document, and no attempt is made to parse the data for symbol substitution. The BINARY option allows the application to insert blocks of data that must not undergo conversion to a client code page when the data is sent. Use the LENGTH option to specify the length of this buffer.

BOOKMARK(*name***)**
specifies a bookmark to be inserted into the document. A bookmark is a symbolic name which identifies an insertion point in the document. The name can be up to 16 characters in length, and must not contain any imbedded spaces.

DOCSIZE(*data-value***)**
specifies a binary fullword area to be updated with the current size of the document in bytes. This is the maximum size of the buffer needed to contain a copy of the document when a RETRIEVE command is issued.

DOCTOKEN(*data-area***)**
specifies the 16-byte binary token of the document into which data is to be inserted.

FROM(data-area)
specifies that a buffer of data supplied by the application is to be inserted into the document. The data content can be a template or a document that was previously created and retrieved. If the data is a template, symbol substitution takes place where the symbols exist in the symbol table. If the data is a previously retrieved document, the conversion and bookmark tags which were inserted during the retrieval will be removed from the content and stored in the internal form required by the API commands. Note that symbol substitution will not be attempted on any unresolved symbols contained in a retrieved document. Use the LENGTH option to specify the length of this buffer.

FROMDOC(*data-area***)**
specifies the binary token of a document (see the DOCTOKEN option) whose contents are copied to the insertion point of the target document. All bookmarks and conversion tags are copied to the target document. The symbol table is not copied.

HOSTCODEPAGE(name)
specifies the symbolic name (see the DOCTOKEN option) of the host code page that the data being added is encoded in. This option applies to the TEXT, SYMBOL and TEMPLATE options only. The name must be eight characters long; if it is shorter than eight characters, it must be padded on the right with blanks.

The standard CICS form of a host code page name consists of the code page number (or more generally CCSID) written using 3 to 5 decimal digits as necessary then padded with trailing spaces to 8 characters. For code page 37, which is fewer than 3 digits, the standard form is 037. CICS now also accepts any decimal number of up to 8 digits (padded with trailing spaces) in the range 1 to 65535 as a code page name, even if it is not in the standard form.

Note that the HOSTCODEPAGE parameter must specify an EBCDIC-based code page if any symbol processing will be required, as the delimiters used for symbol and symbol list processing are assumed to be in EBCDIC.

LENGTH(data-value)
specifies the length, as a fullword binary value, of the buffer containing the TEXT, BINARY or FROM data.

When the DOCUMENT INSERT command follows a DOCUMENT RETRIEVE command, without the use of the DATAONLY option, and the retrieved document is being inserted using the FROM option, the LENGTH specified must be equal to the length of the retrieved document.

SYMBOL(*name***)**
specifies the 32-byte name of a symbol in the symbol table. The data associated with the symbol in the symbol table is inserted, but not the symbol itself. Note that when data associated with a symbol has been inserted into a document, you cannot change that data in the document that is being composed. If you set a different value for the symbol, the new value will be used the next time

that symbol is inserted into a document. Your change will not affect the value that has already been inserted into the document.

TEMPLATE(*name***)**
specifies the 48-byte name of a template. The template must be defined to CICS using RDO. If the name is less than 48 bytes, it must be padded on the right with blanks. The current values of any symbols are substituted into the template.

Note: When a template containing symbols has been inserted into a document, you cannot change the substituted values of those symbols in the document that is being composed. If you set different values for the symbols, the new values will be used the next time that the template is inserted into a document. Your changes will not affect the values that have already been inserted into the document.

TEXT(*data-area***)**
specifies a buffer of data to be inserted into the document. The data is copied unchanged to the insertion point in the document, and no attempt is made to parse the data for symbol substitution. When the document is sent, it is marked as requiring conversion to the client code page. Use the LENGTH option to specify the length of this buffer.

TO(*name***)**
specifies the symbolic name of a bookmark identifying the end position of an overlay operation. Data between the bookmarks identified by the AT and TO operands is deleted, and new data is inserted in its place. It is possible to delete data between two bookmarks by specifying a null string on the TEXT or BINARY option with a LENGTH of zero.

Conditions

14 DUPREC
The bookmark has already been defined.

16 INVREQ
RESP2 values are:

0 The bookmark specified on the TO option appears before the bookmark specifed on the AT bookmark.

1 The retrieved document specified on the FROM option is not in a valid RETRIEVE format.

2 The bookmark name on the BOOKMARK option is invalid.

LENGERR
RESP2 value:

1 The value specified for LENGTH is invalid. The value is negative.

70 NOTAUTH
The command has failed a resource security check. (If the NOTAUTH condition is not handled, applications that receive it may abend with code AEY7.)

Note that the EXEC CICS DOCUMENT commands reference document templates using the 48-character name of the template (as specified in the TEMPLATENAME attribute of the DOCTEMPLATE resource definition). However, security checking for the commands uses the name of the DOCTEMPLATE resource definition that corresponds to the TEMPLATENAME

attribute. If resource security checking is in place, the user ID for the transaction must have READ access to this DOCTEMPLATE resource definition.

RESP2 value:

101 The user ID for the transaction does not have READ access to the DOCTEMPLATE resource definition for the document template named by the TEMPLATE option.

13 NOTFND
One of the following documents or templates could not be found, or its name was incorrect.

RESP2 values:

1 The document specified on the DOCUMENT option.

2 The document specified on the FROMDOC option.

3 The template specified on the TEMPLATE option.

4 The document specified on the SYMBOL option.

5 The document specified on the AT option.

6 The document specified on the TO option.

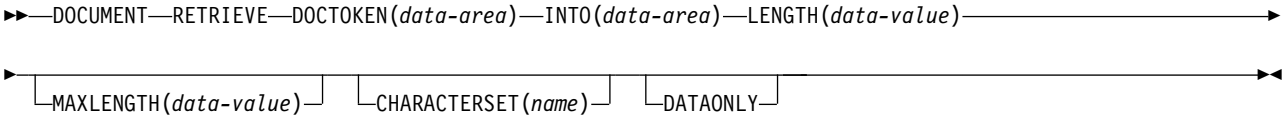
7 The document specified on the HOSTCODEPAGE option.

117 TEMPLATERR
An invalid #set, #include or #echo command has been encountered while processing the supplied template data, or the CICS region does not have the correct level of authority to access the zFS file of this template. RESP2 contains either a zero (if the maximum of 32 levels of embedded templates is exceeded), or the offset of the invalid command.

DOCUMENT RETRIEVE

Copy a document into the application's own buffer.

DOCUMENT RETRIEVE



Conditions: INVREQ, LENGERR, NOTFND

This command is threadsafe.

Description

DOCUMENT RETRIEVE allows the application to obtain a copy of the document in its own buffer, which it can then manipulate directly. The document is managed by CICS, and the application does not have direct access to the buffer containing the contents of the document. The document exists only for the duration of the current transaction, so the application must retrieve the document and store it if the

document is to exist over transaction boundaries. The retrieved document can be used as a basis for a new document by using the FROM option of the **DOCUMENT CREATE** command.

When the document is retrieved, CICS inserts tags into the document contents to identify the bookmarks and to delimit the blocks that do not require code page conversion. To request a copy without tags, specify DATAONLY. The extracted document can also be converted into a single client code page by using the CHARACTERSET option.

The **DOCUMENT CREATE** and **DOCUMENT INSERT** commands return a DOCSIZE value. This value is the maximum size of the buffer needed to contain a copy of the document in its original code page (including control information), when the RETRIEVE command is issued. However, when the CHARACTERSET option specifies an encoding that requires more bytes than the original EBCDIC data (for example, UTF-8), the maximum size might not be large enough to store the converted document. If the DOCSIZE value is used for the buffer size in this case, the program should be prepared to handle a LENGERR condition and acquire a new buffer using the size returned in the LENGTH parameter. Alternatively, you can determine the actual document length before allocating the buffer by issuing a **DOCUMENT RETRIEVE** with a dummy buffer and a MAXLENGTH of zero, then handling the LENGERR condition and using the returned LENGTH value.

Options

CHARACTERSET(*name***)**
specifies the name of the client character set to which the data should be converted. The name can be up to 40 characters in length (if it is shorter than 40 characters, it must be padded on the right with blanks).

This parameter replaces the CLNTCODEPAGE parameter, which is supported for upgrade purposes only.

CLNTCODEPAGE(*name***)**
This option is supported for upgrade purposes only. CHARACTERSET replaces it. The action taken by CICS is the same for both keywords.

DATAONLY
specifies that the data should be retrieved without any imbedded tags.

DOCTOKEN(*data-area***)**
specifies the 16-byte binary token of the document to be retrieved.

INTO(*data-area***)**
specifies the buffer that is to contain the copy of the document content.

LENGTH(*data-value***)**
specifies the length, as a fullword binary value, of the amount of data being returned to the application. If the document is truncated, this is the exact length required to return the whole document.

MAXLENGTH(*data-value***)**
specifies the length, as a fullword binary value, of the maximum amount of data the buffer can receive.

Conditions

16 INVREQ
RESP2 values:

11 An invalid or unsupported combination of code pages was specified.

12 An error occurred during CCSID conversion and the conversion could not be completed. For example, a piece of text or a symbol value ended part of the way through a multi-byte character.

22 LENGERR
RESP2 values:

1 MAXLENGTH is less than zero. The document size is not returned in the LENGTH field.

2 The length of the receiving buffer is zero, or is too short to contain the document contents. The document is truncated and the exact length required is returned in the LENGTH field.

13 NOTFND
RESP2 values:

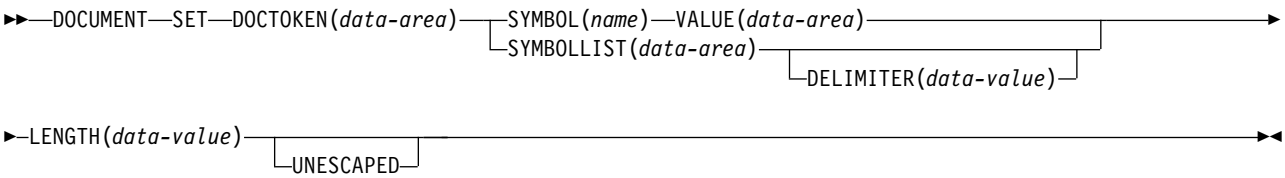
1 The document has not been created, or the name is incorrectly specified.

7 The specified character set cannot be found.

DOCUMENT SET

Add symbols and values to symbol table.

DOCUMENT SET



Conditions: INVREQ, LENGERR, NOTFND, SYMBOLERR

This command is threadsafe.

Description

DOCUMENT SET allows the application to add symbols and their associated values to the symbol table. If the symbol being added already exists in the table, it is replaced by the new definition.

Note:

1. When a template containing symbols has been inserted into a document, you cannot change the substituted values of those symbols in the document that is being composed. If you set different values for the symbols, the new values will be used the next time that the template is inserted into a document. Your changes will not affect the values that have already been inserted into the document.
2. If you insert a template before the symbols contained in it are set, the symbols will never be substituted. This can occur if you create a document from a template without specifying a symbol list.

Options

DELIMITER(data-value)
specifies an optional 1–byte value used to delimit symbol name-value pairs in the SYMBOLLIST buffer. If this option is not specified, the value defaults to an ampersand. Certain delimiter values (such as the space character) are disallowed, and all of these cause an INVREQ condition on the command if used. They are listed in Rules for specifying symbols and symbol lists.

If this option is used, the application must ensure that the DELIMITER does not appear in any symbol value in the SYMBOLLIST buffer. For this reason, the application should not use alphanumeric and other printable characters as the DELIMITER value.

DOCTOKEN(*data-area*)
specifies the 16-byte binary token of the document that owns the symbol table.

LENGTH(*data-value*)
specifies the length, as a fullword binary value, of the buffer containing the data value associated with the symbol, or the length of the buffer containing the symbol list when the SYMBOLLIST option is used.

SYMBOL(*name*)
specifies the name of the symbol that is to be added to the table. The name can be 1 to 32 characters in length with no embedded spaces. Rules for specifying symbols and symbol lists lists the rules which apply when specifying the name of a symbol. If you want to define more than one symbol in the same command, use the SYMBOLLIST option instead.

SYMBOLLIST(*data-area*)
specifies a buffer which contains a symbol list. Use the LENGTH option to specify the length of this buffer. A symbol list is a character string consisting of one or more symbol definitions separated by ampersands. Each symbol definition consists of a name, an equals sign, and a value.Here is an example of a symbol list:

```
applid=IYCQ&jobname=test
```

By default, symbols in the symbol list are separated by the & character, but you can override this by using the DELIMITER keyword to specify a different symbol separator. Rules for specifying symbols and symbol lists lists the rules which apply when setting symbols using a SYMBOLLIST.

UNESCAPED
prevents CICS from unescaping symbol values contained in the SYMBOLLIST buffer. If this option is used, plus signs are not converted to spaces, and sequences such %2B are not converted to single byte values.

The UNESCAPED option does not allow you to include the character that you have used as the symbol separator within a symbol value in a symbol list. If you want to use the UNESCAPED option, choose a symbol separator that will never be used within a symbol value. Alternatively, you can use the SYMBOL and VALUE options to specify symbol values that contain the character you have used as the symbol separator, because the symbol separator has no special meaning when used in the VALUE option.

VALUE(*data-area*)
specifies an area containing the value to be associated with the SYMBOL. Rules for specifying symbols and symbol lists lists the rules which apply when specifying the value of a symbol.

Conditions

16 INVREQ
RESP2 values:

8 The value specified for DELIMITER is not valid.

LENGERR
RESP2 value:

9 The value specified for symbol list LENGTH is invalid. Value must be between 1 and (16M - 1).

10 The value specified for symbol value LENGTH is invalid. Value must be between 1 and (16M - 1).

13 NOTFND
RESP2 values:

1 The document has not been created or the name is incorrectly specified.

116 SYMBOLERR
a symbol name is invalid. RESP2 values:

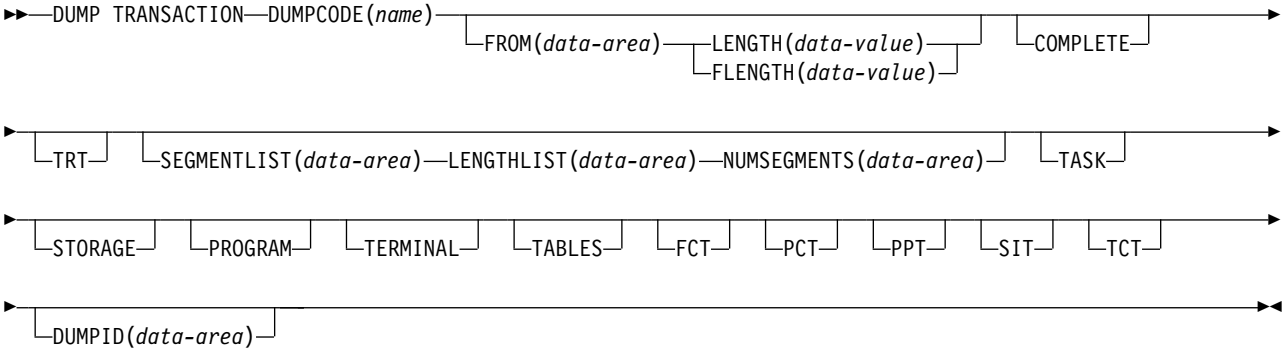
0 SYMBOLLIST was not used.

offset RESP2 contains the offset of the invalid symbol in the list.

DUMP TRANSACTION

Request a transaction dump.

DUMP TRANSACTION



Conditions: INVREQ, IOERR, NOSPACE, NOSTG, NOTOPEN, OPENERR, SUPPRESSED

Description

DUMP TRANSACTION dumps all, a series, or any single main storage area related to a task, any or all of the CICS tables, or all of these together.

Note that if you issue a DUMP TRANSACTION for a DUMPCODE that is defined in the transaction dump table with SYSDUMP, you also get a system dump.

If there is no entry in the system dump table for the specified DUMPCODE, a temporary entry is made. This entry is lost on the next CICS start. The system dump table is described in The dump code options you can specify.

Options

COMPLETE
dumps all main storage areas related to a task, all the CICS tables, and the DL/I control blocks.

DUMPCODE(*name***)**
specifies a name (1–4 characters) that identifies the dump. If the name contains any leading or imbedded blanks, the dump is produced but INVREQ is raised. No entry is added to the system dump table.

If you omit all the options except DUMPCODE, you get the same dump as if you specified TASK, but without the DL/I control blocks.

DUMPID(*data-area***)**
returns a 6- to 9-character dump identifier generated for this particular dump. The format of the identifier is xxxx/yyyy, where *xxxx* represents the **dump run number**, yyyy is the **dump count**, and the slash (/) symbol is a separator character. The dump identifier is generated as follows:

Dump run number
A number in the range 1 to 9999. (Leading zeros are not used for this number, which is why the dump id can vary from 6 to 9 characters.) The dump run number begins at 1 when you first start CICS with a newly-initialized local catalog, and is incremented by 1 each time you restart CICS. The dump run number is saved in the local catalog when you perform a normal shutdown, but is reset if you start CICS with a START=INITIAL or START=COLD system initialization parameter.

Dump count
A number in the range 0001 through 9999. (Leading zeros are required in the dump id.) This is the number assigned to the dump in this run of CICS, starting at 0001 for the first dump, and incremented by 1 with each dump taken.

FCT
dumps the file control table, which contains FILE resource definitions.

FLENGTH(*data-value***)**
specifies the length (fullword binary value) of the storage area (specified in the FROM option) that is to be dumped. The maximum length that you can specify is 16 777 215 bytes.

FLENGTH and LENGTH are mutually exclusive.

FROM(*data-area***)**
dumps the specified data area, which must be a valid area; that is, storage allocated by the operating system within the CICS region. In addition, the following areas are dumped:

- Task control area (TCA) and, if applicable, the transaction work area (TWA).
- Common system area (CSA), including the user's portion of the CSA (CWA).
- If TRAN is specified for the TRTRANTY SIT parameter, only the trace entries associated with the current task are formatted. If TRTRANTY=ALL is specified, the entire internal trace table is formatted. This applies only when the CICS trace facility is active.

- Either the terminal control table terminal entry (TCTTE) or the transient data queue definition associated with the requesting task.

Whenever the TCTTE is dumped, the terminal control table user area (if any) and the message control blocks (if any) associated with the TCTTE are dumped. The latter are used by basic mapping support.

LENGTH(*data-value***)**
specifies the length (halfword binary) of the data area specified in the FROM option. For a description of a safe upper limit, see "LENGTH options in CICS commands" on page 10.

LENGTH and FLENGTH are mutually exclusive.

LENGTHLIST(*data-area***)**
specifies a list of 32-bit binary values showing the lengths of the storage areas to be dumped. This corresponds to the list of segments specified in the SEGMENTLIST option. You must use both the SEGMENTLIST and NUMSEGMENTS options when you use the LENGTHLIST option.

NUMSEGMENTS(*data-area***)**
specifies the number (fullword binary) of areas to be dumped. You must use both the SEGMENTLIST and LENGTHLIST options when you use the NUMSEGMENTS option.

PCT
formats a summary of each installed transaction resource definition.

PPT
formats a summary of each installed program resource definition.

PROGRAM
specifies that program storage areas associated with this task are to be dumped, as follows:

- Task control area (TCA) and, if applicable, the transaction work area (TWA)
- Common system area (CSA), including the user's portion of the CSA (CWA)
- Trace entries relating to the task written to the internal trace table (only when the CICS trace facility is active)
- All program storage areas containing user-written application programs active on behalf of the requesting task
- Register save areas (RSAs) indicated by the RSA chain off the TCA
- Either the terminal control table terminal entry (TCTTE) or the transient data queue definition associated with the requesting task.

Whenever the TCTTE is dumped, the terminal control table user area (if any) and the message control blocks (if any) associated with the TCTTE are dumped.

SEGMENTLIST(*data-area***)**
specifies a list of addresses, which are the starting points of the segments to be dumped. Each segment is a task-related storage area. You must use both the NUMSEGMENTS and LENGTHLIST options when you use the SEGMENTLIST option.

SIT
dumps the system initialization table.

STORAGE
specifies that storage areas associated with this task are to be dumped, as follows:

- Task control area (TCA) and, if applicable, the transaction work area (TWA)
- Common system area (CSA), including the user's portion of the CSA (CWA)
- Trace entries relating to the task written to the internal trace table (only when the CICS trace facility is active)
- All transaction storage areas
- Either the terminal control table terminal entry (TCTTE) or the transient data queue definition associated with the requesting task

Whenever the TCTTE is dumped, the terminal control table user area (if any) and the message control blocks (if any) associated with the TCTTE are dumped.

TABLES
dumps the FCT (file control table, containing FILE resource definitions), PCT (program control table, containing TRANSACTION resource definitions), PPT (processing program table, containing PROGRAM resource definitions), SIT (system initialization table, containing CICS system initialization parameters), and the TCT (terminal control table, containing TERMINAL resource definitions).

TASK
specifies that storage areas associated with this task are to be dumped, as follows:

- A summary of the transaction environment associated with this task
- Common system area (CSA), including the user's portion of the CSA (CWA)
- Trace entries relating to the task written to the internal trace table (only when the CICS trace facility is active)
- All program storage areas containing user-written application programs active on behalf of the requesting task
- All transaction storage areas
- Either the terminal control table terminal entry (TCTTE) or the transient data queue definition associated with the requesting task
- Register save areas (RSAs) indicated by the RSA chain off the TCA
- All terminal input/output areas (TIOAs) chained off the terminal control table terminal entry (TCTTE) for the terminal associated with the requesting task
- DL/I control blocks

Whenever the TCTTE is dumped, the terminal control table user area (if any) and the message control blocks (if any) associated with the TCTTE are dumped.

TCT
dumps the terminal control table.

TERMINAL
specifies that storage areas associated with the terminal are to be dumped, as follows:

- Task control area (TCA) and, if applicable, the transaction work area (TWA)
- Common system area (CSA), including the user's portion of the CSA (CWA)
- Trace entries relating to the task written to the internal trace table (only when the CICS trace facility is active)

- All terminal input/output areas (TIOAs) chained off the terminal control table terminal entry (TCTTE) for the terminal associated with the requesting task as long as the request is not a write, or storage freezing is on for the task or the terminal
- Either the terminal control table terminal entry (TCTTE) or the transient data queue definition associated with the requesting task.

Whenever the TCTTE is dumped, the terminal control table user area (if any) and the message control blocks (if any) associated with the TCTTE are dumped. The latter are used by basic mapping support.

TRT
dumps the trace entries relating to the task written to the internal trace table.

Conditions

16 INVREQ
RESP2 values:

13 An incorrect DUMPCODE is specified. DUMPCODE contains unprintable characters, or leading or imbedded blanks.

The dump is produced but no entry is added to the system dump table.

Default action: terminate the task abnormally.

17 IOERR
RESP2 values:

9 The SDUMP process is not authorized.

10 An error occurred during system dumping.

13 The CICS routine issuing the SDUMP is unable to establish a recovery routine (FESTAE).

Default action: terminate the task abnormally.

18 NOSPACE
RESP2 values:

4 The transaction dump is incomplete due to lack of space.

Default action: terminate the task abnormally.

42 NOSTG
RESP2 values:

5 CICS has run out of working storage.

Default action: terminate the task abnormally.

19 NOTOPEN
RESP2 values:

6 The current CICS dump data set is not open.

Default action: terminate the task abnormally.

87 OPENERR
RESP2 values:

7 There is an error on opening, closing, or writing to the current CICS dump routine.

Default action: terminate the task abnormally.

72 SUPPRESSED
RESP2 values:

1 The transaction dump is suppressed by MAXIMUM in table.

2 The transaction dump is suppressed by NOTRANDUMP in table.

3 The transaction dump is suppressed by a user exit program.

Default action: terminate the task abnormally.

Examples

The following example shows how to request a dump of all the task-related storage areas, the terminal control table, and a specified data area:

```
EXEC CICS DUMP TRANSACTION
     DUMPCODE('name')
     FROM(data-area)
     LENGTH(data-value)
```

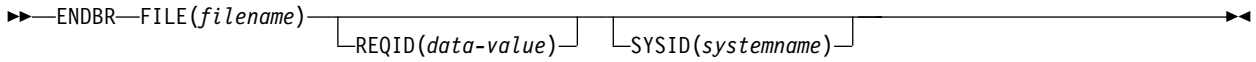
This second example (written in PL/I) shows you a case in which five task-related storage areas are dumped:

```
DCL storage_address(5)   POINTER,
    storage_length(5)    FIXED BIN(31),
    nsegs                FIXED BIN(31);
storage_address(1) = ADDR(areal);
storage_length(1)  = CSTG(areal);
:
:
nsegs = 5;
EXEC CICS DUMP TRANSACTION
     DUMPCODE('name')
     SEGMENTLIST(storage_address)
     LENGTHLIST(storage_length)
     NUMSEGMENTS(nsegs);
```

ENDBR

End the browse of a file.

ENDBR



Conditions: FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, NOTAUTH, SYSIDERR

This command is threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over an IPIC connection to a remote CICS region.
- Defined as either local VSAM or RLS.

This command is not threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over a non-IPIC connection.
- Defined as a shared data table, coupling facility data table, or BDAM file.

Description

ENDBR ends a browse on a file or data table on a local or a remote CICS region.

The UPDATE option is available within browse so we recommend that you use this because otherwise you would need to issue an ENDBR command before using READ UPDATE to avoid self deadlock abends. We recommend issuing an ENDBR before syncpoint for similar reasons.

If STARTBR was not successful, you do not need to issue ENDBR.

Options

FILE(*filename***)**
Specifies the name of the file being browsed.

If SYSID is specified, the data set to which this file refers is assumed to be on a remote system irrespective of whether the name is defined to CICS. Otherwise, the resource definition for the file is used to find out whether the data set is on a local or a remote system.

REQID(*data-value***)**
Specifies a unique (halfword binary value) request identifier for a browse, used to control multiple browse operations on a data set. If this option is not specified, a default value of zero is assumed.

SYSID(*systemname***)**
Specifies the name (1–4 characters) of the system the request is directed to.

Conditions

12 FILENOTFOUND
RESP2 values:

1 The name referred to in the FILE option is not defined to CICS.

Default action: terminate the task abnormally.

21 ILLOGIC (VSAM)
RESP2 values:

110 A VSAM error occurs that is not in one of the other CICS response categories.

See EIBRCODE in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

35 The REQID, SYSID, or file name does not match that of any successful STARTBR command.

Default action: terminate the task abnormally.

17 IOERR
RESP2 values:

120 There is an I/O error during the file control operation. An I/O error is any unusual event that is not covered by a CICS condition.

For VSAM files, IOERR usually indicates a hardware error. Further information is available in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

54 ISCINVREQ
RESP2 values:

70 The remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

70 NOTAUTH
RESP2 values:

101 A resource security check has failed on FILE(filename).

Default action: terminate the task abnormally.

53 SYSIDERR
RESP2 values:

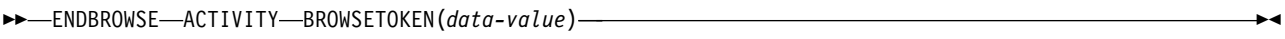
130 The SYSID option specifies a name that is neither the local system nor a remote system that is defined by a CONNECTION or IPCONN definition. SYSIDERR also occurs when the link to the remote system is known but unavailable. In the case of an IPCONN, SYSIDERR occurs if the link is known but either the local or remote systems do not support file control commands that are function shipped using IP interconnectivity.

Default action: terminate the task abnormally.

ENDBROWSE ACTIVITY

End a browse of the child activities of a BTS activity, or of the descendant activities of a BTS process.

ENDBROWSE ACTIVITY



Conditions: ILLOGIC, TOKENERR

Description

ENDBROWSE ACTIVITY ends a browse of the child activities of a BTS activity (or of the descendant activities of a BTS process), and invalidates the browse token.

Options

BROWSETOKEN(data-value)
specifies, as a fullword binary value, the browse token to be deleted.

Conditions

21 ILLOGIC
RESP2 values:

1 The value specified in the BROWSETOKEN option matches a current browse token, but not one that is being used for an activity browse.

112 TOKENERR
RESP2 values:

3 The browse token is not valid.

ENDBROWSE CONTAINER (BTS)

End a browse of the containers associated with a BTS activity or process.

ENDBROWSE CONTAINER



Conditions: ILLOGIC, TOKENERR

Description

ENDBROWSE CONTAINER ends a browse of the containers associated with a BTS activity or process, and invalidates the browse token.

Options

BROWSETOKEN(data-value)
specifies, as a fullword binary value, the browse token to be deleted.

Conditions

21 ILLOGIC
RESP2 values:

1 The value specified in the BROWSETOKEN option matches a current browse token, but not one that is being used for a browse of containers.

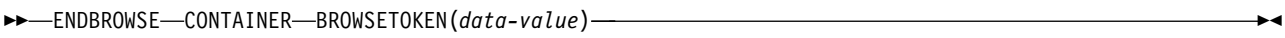
112 TOKENERR
RESP2 values:

3 The browse token is not valid.

ENDBROWSE CONTAINER (CHANNEL)

End a browse of the containers associated with a channel.

ENDBROWSE CONTAINER



Conditions: ILLOGIC, TOKENERR

Description

ENDBROWSE CONTAINER (CHANNEL) ends a browse of the containers associated with a channel, and invalidates the browse token.

Options

BROWSETOKEN(data-value)
specifies, as a fullword binary value, the browse token to be deleted.

Conditions

21 ILLOGIC
RESP2 values:

1 The value specified in the BROWSETOKEN option matches a current browse token, but not one that is being used for a browse of containers.

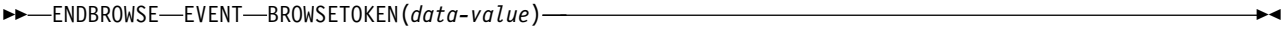
112 TOKENERR
RESP2 values:

3 The browse token is not valid.

ENDBROWSE EVENT

End a browse of the events known to a BTS activity.

ENDBROWSE EVENT



Conditions: TOKENERR

Description

ENDBROWSE EVENT ends a browse of the events that are within the scope of a BTS activity, and invalidates the browse token.

Options

BROWSETOKEN(data-value)
specifies, as a fullword binary value, the browse token to be deleted.

Conditions

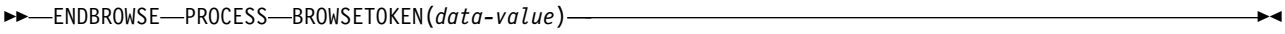
112 TOKENERR
RESP2 values:

3 The browse token is not valid.

ENDBROWSE PROCESS

End a browse of processes of a specified type within the CICS business transaction services system.

ENDBROWSE PROCESS



Conditions: ILLOGIC, TOKENERR

Description

ENDBROWSE PROCESS ends a browse of the processes of a specified type within the CICS business transaction services system, and invalidates the browse token.

Options

BROWSETOKEN(data-value)
specifies, as a fullword binary value, the browse token to be deleted.

Conditions

21 ILLOGIC
RESP2 values:

1 The value specified in the BROWSETOKEN option matches a current browse token, but not one that is being used for a process browse.

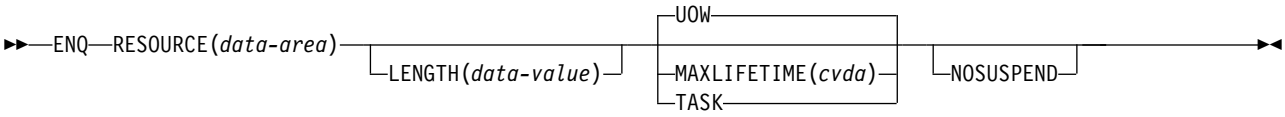
112 TOKENERR
RESP2 values:

3 The browse token is not valid.

ENQ

Schedule use of a resource by a task (enqueue).

ENQ



Conditions: ENQBUSY, INVREQ, LENGERR

This command is threadsafe if it is defined as LOCAL. It is non-threadsafe if it is defined as GLOBAL.

Description

ENQ causes further execution of the task issuing the ENQ command to be synchronized with the availability of the specified resource; control is returned to the task when the resource is available.

A resource in the context of this command is any string of 1–255 bytes, established by in-house standards, to protect against conflicting actions between tasks, or to cause single threading within a program.

If a task enqueues on a resource but does not dequeue from it, CICS automatically releases the resource during syncpoint processing (including DL/I, PCB, and TERM calls), or when the task is terminated. Option UOW forces the dequeue at the end of a unit of work (UOW). Option TASK forces the dequeue at the end of a task. If there are several units of work in a task, the enqueue carries over the UOWs.

If more than one ENQ command is issued for the same resource by a given task, the resource remains owned by that task until the task issues a matching number of DEQ commands.

The resource to be enqueued on must be identified by one of the following methods:

- Specifying a data area that is the resource. It is the location (address) of the data area in storage that matters, not its contents.
- Specifying a data area that contains a unique argument (for example, an employee name) that represents the resource. It is the contents of the data value that matters, not its location. LENGTH is required; the presence of the LENGTH option tells CICS to enqueue on the contents of the data value.

When an EXEC CICS ENQ (or DEQ) command is issued for a resource whose name matches that of an installed ENQMODEL resource definition, CICS checks the value of the ENQSCOPE attribute to determine whether the scope is local or sysplex-wide. If the ENQSCOPE is non-blank, CICS treats the ENQ or DEQ as sysplex-wide, and passes a queue name and the resource name to z/OS global resource serialization to manage the enqueue. If the ENQSCOPE attribute is left blank (the default value), CICS treats the ENQ as local to the issuing CICS region. If no ENQMODEL matches the resource name, the scope of the ENQ command is local. For more information on ENQMODEL resources, see ENQMODEL resources.

Resource unavailability

If a resource is not available when ENQ is issued, the application program is suspended until it becomes available. However, if the NOSUSPEND option has been specified and the resource is unavailable, the ENQBUSY condition is raised, as it is also raised if you have an active HANDLE condition. This allows the application program to handle the case of resource unavailability (by HANDLE CONDITION ENQBUSY) without waiting for the resource to become available.

Options

LENGTH(*data-value***)**
specifies as a halfword binary value the length of the resource to be enqueued on. The value must be in the range 1 through 255; otherwise, the LENGERR condition occurs. If the LENGTH option is specified in an ENQ command, it must also be specified in the DEQ command for that resource, and the values of these options must be the same. You must specify LENGTH when using the method that specifies a data value containing a unique argument, but not for the method that specifies a data area as the resource. It is the presence or absence of LENGTH that tells CICS which method you are using.

MAXLIFETIME(*cvda*)
specifies the duration of the ENQ to be automatically released by CICS. CVDA values are:

UOW The duration of the ENQ is a unit of work. Examples are a syncpoint rollback or syncpoint, if the application does not issue a DEQ before the unit of work ends. This is the default value.

Note: For compatibility, a CVDA value of LUW is also supported.

TASK
The duration of the ENQ is a task. The enqueue carries over the units of work within the task. Use MAXLIFETIME(TASK) with great care because other tasks issuing ENQ commands on the same resource could be suspended until the end of this task.

There are two ways to code this option.

- You can assign a CVDA value with the translator routine DFHVALUE. This allows you to change a CVDA value in the program. For example:

```
MOVE DFHVALUE(UOW) TO AREA-A
EXEC CICS ENQ RESOURCE(RESNAME)
             MAXLIFETIME(AREA-A)
```

- If the required action is always the same, you can declare the value directly. For example:
  or

```
EXEC CICS ENQ RESOURCE(RESNAME) UOW


EXEC CICS ENQ RESOURCE(RESNAME) TASK
```

NOSUSPEND
specifies that the application program is not to be suspended if the resource on the ENQ command is unavailable, but the ENQBUSY condition occurs.

Note, however, that if a HANDLE CONDITION for ENQBUSY is active when the command is executed, action, control is passed to the user label supplied in the HANDLE CONDITION. This takes precedence over the NOSUSPEND option but is, of course, negated by either NOHANDLE or RESP.

RESOURCE(*data-area*)
identifies the resource to be enqueued on by:

- Specifying an area whose address represents the resource.
- Specifying a variable that contains the resource (an employee name, for example). In this case, you must use the LENGTH option.

Conditions

55 ENQBUSY
occurs when an ENQ command specifies a resource that is unavailable and the NOSUSPEND option is specified, or there is an active HANDLE CONDITION ENQBUSY.

If the NOSUSPEND option is not specified, and the ENQ command specifies a resource that is unavailable, the application program is suspended and the ENQBUSY condition is not raised.

Default action: ignore the condition.

16 INVREQ
RESP2 values: CVDA values are:

2 The MAXLIFETIME option is set with an incorrect CVDA.

Default action: terminate the task abnormally.

22 LENGERR
RESP2 values:

1 The value specified for the LENGTH option is outside the range 1 through 255.

Default action: terminate the task abnormally.

Examples

Two tasks, enqueuing on the same resource and specifying a data area that is the resource, must refer to the same location in storage. They could both, for example, refer to the same location in the CWA.

```
EXEC CICS ENQ
     RESOURCE(RESNAME)
```

Two tasks, enqueuing on the same resource and specifying a data area that contains a unique argument, can refer to the same location or to different locations, but the contents of the locations must be the same. The length must be supplied in the LENGTH option.

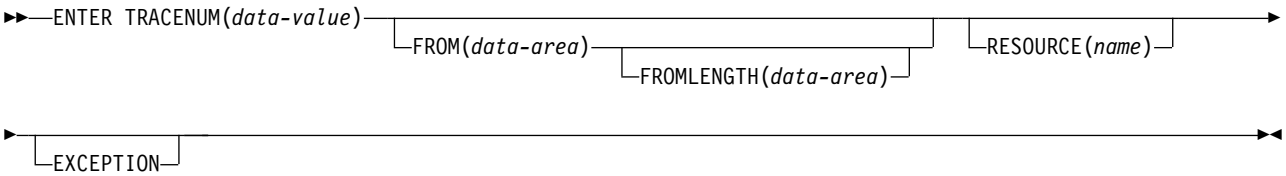
```
EXEC CICS ENQ
     RESOURCE(SOCSECNO)
     LENGTH(9)
```

The two methods cannot be combined. If one task uses the LENGTH option, and the other task does not, CICS regards the enqueues with and without the LENGTH option as different types of enqueues, and the tasks are not serialized.

ENTER TRACENUM

Write a trace entry.

ENTER TRACENUM



Conditions: INVREQ, LENGERR

This command is threadsafe.

Description

The ENTER TRACENUM command makes a trace entry in the currently active trace destinations. CICS writes the trace entry only if the master and user trace

flags are on, unless you specify the EXCEPTION option, in which case a user trace entry is always written, even if the master and user trace flags are off. Exception trace entries are always written to the internal trace table (even if internal tracing is set off), but they are written to other destinations only if they are active.

You can use the exception trace option in an application program to write a trace entry when the program detects an exception or abnormal condition. To do this, include an ENTER TRACENUM(data-value) EXCEPTION command in your program's exception or abnormal condition error-handling routine.

To write an exception trace entry in an error situation when an application program has given up control, you can issue an ENTER TRACENUM(data-value) EXCEPTION command from a user-written program error program (PEP). See Writing a program error program for programming information about modifying the DFHPEP program.

Note: ENTER TRACENUM replaces the earlier ENTER TRACEID command, which is still supported for compatibility with releases of CICS earlier than Version 3. You should use ENTER TRACENUM for all new programs, and whenever you apply maintenance to old programs.

For information about the trace entry format, see CICS trace.

Options

EXCEPTION
specifies that CICS is to write a user exception trace entry. The EXCEPTION option overrides the master user trace flag, and CICS writes the trace entry even if the user trace flag is off. Exception trace entries are identified by the characters *EXCU when the trace entries are formatted by the trace utility program. See CICS exception tracing for more information about user exception trace entries.

FROM(*data-area***)**
specifies a data area whose contents are to be entered into the data field of the trace table entry. If you omit the FROM option, two fullwords of binary zeros are passed.

FROMLENGTH(*data-area***)**
specifies a halfword binary data area containing the length of the trace data, in the range 0–4000 bytes. If FROMLENGTH is not specified, a length of 8 bytes is assumed.

RESOURCE(*name***)**
specifies an 8-character name to be entered into the resource field of the trace table entry.

TRACENUM(*data-value***)**
specifies the trace identifier for a user trace table entry as a halfword binary value in the range 0 through 199.

Conditions

16 INVREQ
RESP2 values:

1 TRACENUM is outside the range 0 through 199.

2 There is no valid trace destination.

3 The user trace flag is set OFF and EXCEPTION has not been specified.

Default action: terminate the task abnormally.

22 LENGERR

RESP2 values:

4 FROMLENGTH is outside the range 0 through 4000.

Default action: terminate the task abnormally.

Examples

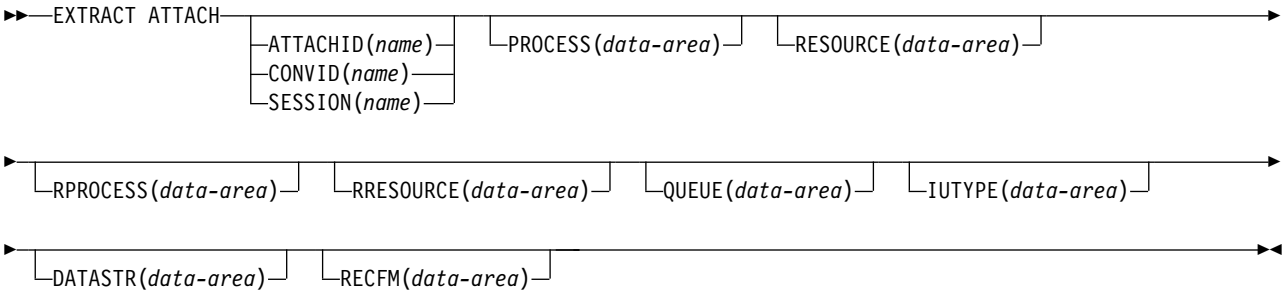
The following COBOL example shows how to write a user trace entry with a trace identifier of 123, with trace data from a data area called USER-TRACE-ENTRY:

```
EXEC CICS ENTER TRACENUM(123)
     FROM(USER-TRACE-ENTRY)
END-EXEC.
```

EXTRACT ATTACH (LUTYPE6.1)

Retrieve values from an LUTYPE6.1 attach header.

EXTRACT ATTACH (LUTYPE6.1)



Conditions: CBIDERR, INVREQ, NOTALLOC

Description

EXTRACT ATTACH retrieves a set of values that are held in an attach header control block, or that have been built previously. For the command to retrieve information from a received attach Function Management Header (FMH), EIBATT must have been set during a RECEIVE or CONVERSE command.

Options

ATTACHID(*name***)**

specifies that values are to be retrieved from an attach header control block. The name (1–8 characters) identifies this control block to the local task.

CONVID(*name***)**

identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE

command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

DATASTR(*data-area***)**
corresponds to the data stream profile field, ATTDSP, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no particular significance is given by CICS to the data stream profile field in an attach FMH. For most CICS applications, the option can be omitted.

The value returned in the data area is a halfword binary value. Only the low-order byte is used; the SNA-defined meanings of the bits are as follows:

```
0-7    reserved - must be set to zero
8-11   0000 - user-defined
       1111 - SCS data stream
       1110 - 3270 data stream
       1101 - structured field
       1100 - logical record management
12-15  defined by the user if bits 8-11
       are set to 0000; otherwise reserved
       (must be set to zero)
```

IUTYPE(*data-area***)**
corresponds to the interchange unit field, ATTIU, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no particular significance is attached by CICS to the interchange unit field in an attach FMH. For most CICS applications the option can be omitted.

The value returned in the data area is a halfword binary value. Only the low-order 7 bits are used; the SNA-defined meanings of the bits are as follows:

```
0-10   reserved - must be set to zero
11     0 - not end of multichain interchange unit
       1 - end of multichain interchange unit
12,13  reserved - must be set to zero
14,15  00 - multichain interchange unit
       01 - single-chain interchange unit
       10 - reserved
       11 - reserved
```

PROCESS(*data-area***)**
corresponds to the process name, ATTDPN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, a transaction running in one system can acquire a session to the second system and can identify the transaction to be attached; in the second system the identification is carried in the first chain of data sent across the session.

In general, the first four bytes of data identify the transaction to be attached. However an attach FMH, identifying the transaction to be attached, can be built and sent. The receiving CICS system uses just the first four bytes of the process name as a transaction name.

No significance is attached by CICS to process names in attach FMHs sent in chains of data other than the first.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the process name field in an attach FMH.

QUEUE(*data-area***)**
corresponds to the queue name, ATTDQN, in an attach FMH.

For communication between two CICS systems, no significance is attached by CICS to the queue name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the queue name field in an attach FMH.

RECFM(*data-area***)**
corresponds to the deblocking algorithm field, ATTDBA, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no particular significance is attached by CICS to the deblocking algorithm field in an attach FMH. For most CICS applications, the option can be omitted.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the interchange unit field in an attach FMH.

The value returned in the data area is a halfword binary value. Only the low-order byte is used; the SNA-defined meanings of the bits are as follows:

```
0-7   reserved - must be set to zero
8-15  X'00' - reserved
      X'01' - variable-length
              variable-blocked
      X'02' - reserved
      X'03' - reserved
      X'04' - chain of RUs
      X'05' through X'FF' - reserved
```

RESOURCE(*data-area***)**
corresponds to the resource name, ATTPRN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no significance is attached by CICS to the resource name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the resource name field in an attach FMH.

RPROCESS(*data-area***)**
corresponds to the return process name, ATTRDPN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no significance is attached by CICS to the return process name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the return process name field in an attach FMH.

RRESOURCE(*data-area***)**
corresponds to the return resource name, ATTRPRN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no significance is attached by CICS to the return resource name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the return resource name field in an attach FMH.

SESSION(*name*)
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used.

Conditions

62 CBIDERR
occurs if the requested attach header control block cannot be found.

Default action: terminate the task abnormally.

16 INVREQ
occurs if incorrect data is found.

Default action: terminate the task abnormally.

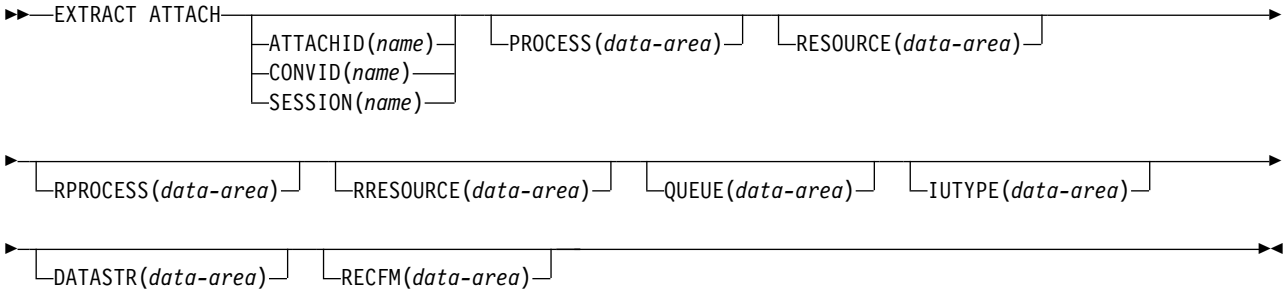
61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

EXTRACT ATTACH (MRO)

Retrieve values from an MRO attach header.

EXTRACT ATTACH (MRO)



Conditions: CBIDERR, INVREQ, NOTALLOC

Description

EXTRACT ATTACH retrieves a set of values that are held in an attach header control block, or that have been built previously. For the command to retrieve information from a received attach Function Management Header (FMH), EIBATT must have been set during a RECEIVE or CONVERSE command.

For more information about MRO and IRC, see Introduction to CICS intercommunication .

Options

ATTACHID(*name*)
specifies that values are to be retrieved from an attach header control block. The name (1–8 characters) identifies this control block to the local task.

CONVID(*name*)
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

DATASTR(*data-area*)
corresponds to the data stream profile field, ATTDSP, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no particular significance is given by CICS to the data stream profile field in an attach FMH. For most CICS applications, the option can be omitted.

The value returned in the data area is a halfword binary value. Only the low-order byte is used; the SNA-defined meanings of the bits are as follows:

```
0-7    reserved - must be set to zero
8-11   0000 - user-defined
       1111 - SCS data stream
       1110 - 3270 data stream
       1101 - structured field
       1100 - logical record management
12-15  defined by the user if bits 8-11
       are set to 0000; otherwise reserved
       (must be set to zero)
```

IUTYPE(*data-area*)
corresponds to the interchange unit field, ATTIU, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no particular significance is attached by CICS to the interchange unit field in an attach FMH. For most CICS applications the option can be omitted. The value returned in the data area is a halfword binary value. Only the low-order 7 bits are used; the SNA-defined meanings of the bits are as follows:

```
0-10   reserved - must be set to zero
11     0 - not end of multichain interchange unit
       1 - end of multichain interchange unit
12,13  reserved - must be set to zero
14,15  00 - multichain interchange unit
       01 - single chain interchange unit
       10 - reserved
       11 - reserved
```

PROCESS(*data-area*)
corresponds to the process name, ATTDPN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, a transaction running in one system can acquire a session to the second system and can identify the transaction to be attached; in the second system the identification is carried in the first chain of data sent across the session.

In general, the first four bytes of data identify the transaction to be attached. However an attach FMH, identifying the transaction to be attached, can be built and sent. The receiving CICS system uses just the first four bytes of the process name as a transaction name. No significance is attached by CICS to process names in attach FMHs sent in chains of data other than the first.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the process name field in an attach FMH.

QUEUE(*data-area***)**
corresponds to the queue name, ATTDQN, in an attach FMH. For communication between two CICS systems, no significance is attached by CICS to the queue name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the queue name field in an attach FMH.

RECFM(*data-area***)**
corresponds to the deblocking algorithm field, ATTDBA, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no particular significance is attached by CICS to the deblocking algorithm field in an attach FMH. For most CICS applications, the option can be omitted.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the interchange unit field in an attach FMH.

The value returned in the data area is a halfword binary value. Only the low-order byte is used; the SNA-defined meanings of the bits are as follows:

```
0-7   reserved - must be set to zero
8-15  X'00' - reserved
      X'01' - variable-length
              variable-blocked
      X'02' - reserved
      X'03' - reserved
      X'04' - chain of RUs
      X'05'through X'FF' - reserved
```

RESOURCE(*data-area***)**
corresponds to the resource name, ATTPRN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no significance is attached by CICS to the resource name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the resource name field in an attach FMH.

RPROCESS(*data-area***)**
corresponds to the return process name, ATTRDPN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no significance is attached by CICS to the return process name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the return process name field in an attach FMH.

RRESOURCE(*data-area***)**
corresponds to the return resource name, ATTRPRN, in an LUTYPE6.1 attach FMH.

For communication between two CICS systems, no significance is attached by CICS to the return resource name in an attach FMH.

For communication between a CICS system and another subsystem, refer to documentation supplied by the subsystem about how to use the return resource name field in an attach FMH.

SESSION(*name*)
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used.

Conditions

62 CBIDERR
occurs if the requested attach header control block cannot be found.

Default action: terminate the task abnormally.

16 INVREQ
occurs if incorrect data is found.

Default action: terminate the task abnormally.

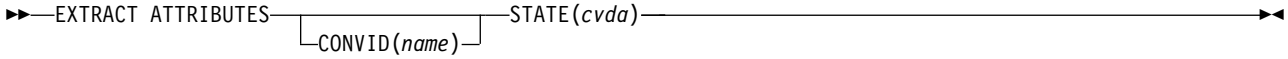
61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

EXTRACT ATTRIBUTES (APPC)

Obtain the state of the APPC conversation.

EXTRACT ATTRIBUTES (APPC)



Conditions: INVREQ, NOTALLOC

Description

EXTRACT ATTRIBUTES extracts conversation state information for APPC mapped conversations.

Options

CONVID(*name*)
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

For compatibility with earlier releases, SESSION is accepted as a synonym for CONVID. New programs should use CONVID.

By default, the principal facility is assumed.

STATE(*cvda*)
gets the state of the transaction program. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application explicitly, or implicitly by default, specified the function-shipping session (its principal facility) on the CONVID option.

also occurs (RESP2 not set) in any of the following situations:

- The command is issued against a CPI-Communications conversation.
- The command is issued against an APPC basic conversation. (A GDS EXTRACT ATTRIBUTES should have been used instead.)

Default action: terminate the task abnormally.

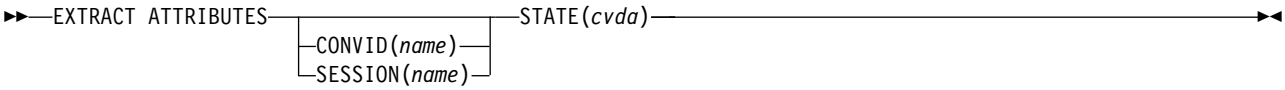
61 NOTALLOC
occurs if the specified CONVID value does not relate to a conversation owned by the application.

Default action: terminate the task abnormally.

EXTRACT ATTRIBUTES (MRO)

Extract attributes from an MRO conversation.

EXTRACT ATTRIBUTES (MRO)



Conditions: INVREQ, NOTALLOC

Description

EXTRACT ATTRIBUTES (MRO) extracts conversation state information for MRO conversations.

Options

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

SESSION(*name***)**
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used.

If both this option and CONVID are omitted, the principal facility for the task is used.

STATE(*cvda***)**
gets the state of the transaction program. The cvda values returned by CICS are:

- ALLOCATED
- FREE
- PENDFREE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application explicitly, or implicitly by default, specified the function-shipping session (its principal facility) on the CONVID option.

also occurs (RESP2 not set) in any of the following situations:

- An incorrect command has been issued for the terminal or LU in use.

Default action: terminate the task abnormally.

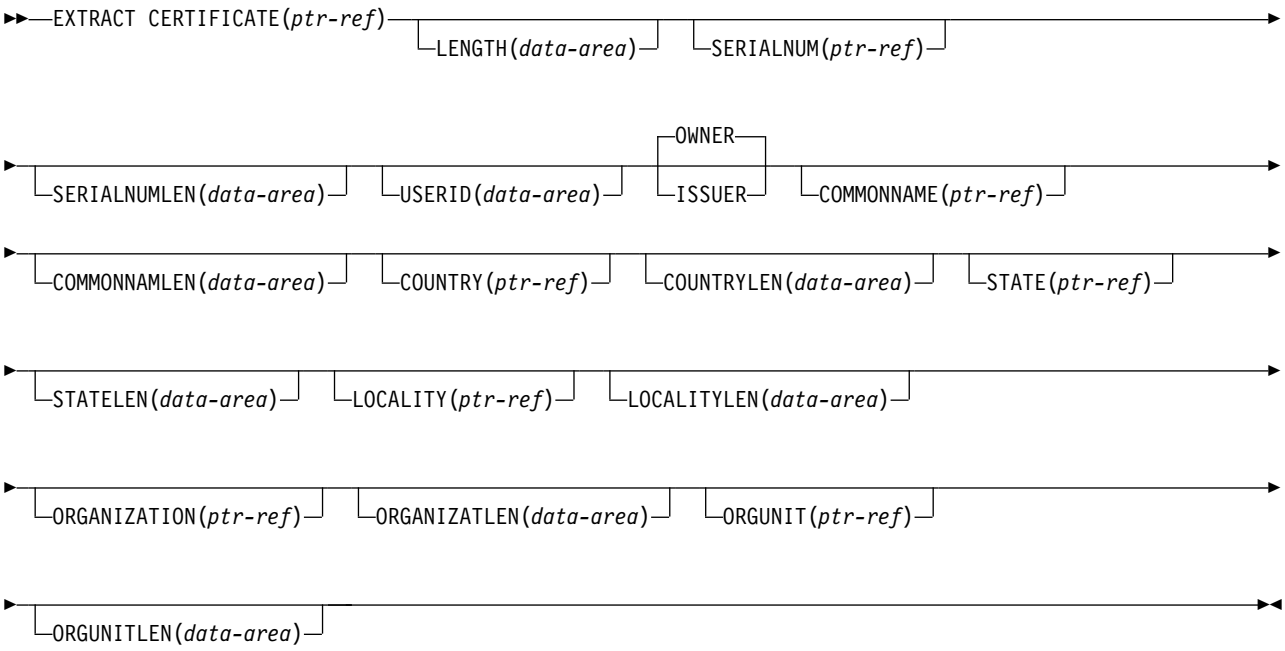
61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

EXTRACT CERTIFICATE

Obtain information from the client certificate received over a TCP/IP service that specified client authentication.

EXTRACT CERTIFICATE



Conditions: INVREQ, LENGERR

This command is threadsafe.

Description

EXTRACT CERTIFICATE allows the application to obtain information from the X.509 certificate that was received from a client during a Secure Sockets Layer (SSL) handshake over a TCPIPSERVICE that specified SSL(CLIENTAUTH). The certificate contains fields that identify the owner (or subject) of the certificate, and fields that identify the certificate authority that issued the certificate. You can select the fields that you require by specifying the OWNER or ISSUER option. You cannot retrieve both OWNER and ISSUER fields with one command.

If you attempt to extract a certificate and there is no certificate to extract, low or zero values are returned for the pointers.

Options

CERTIFICATE(ptr-ref)
Specifies a pointer reference to be set to the address of the full binary certificate received from the client. The pointer reference is valid until the next CICS command or the end of task.

COMMONNAME(ptr-ref)
Specifies a pointer reference to be set to the common name from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

COMMONNAMLEN(data-area)
Specifies a fullword binary data area to be set to the length of the common name from the client certificate.

COUNTRY(ptr-ref)
Specifies a pointer reference to be set to the address of the country from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

COUNTRYLEN(data-area)
Specifies a fullword binary data area to be set to the length of the country from the client certificate.

ISSUER
Indicates that the values returned by this command refer to the certificate authority that issued this certificate.

LENGTH(data-area)
Specifies a fullword binary data area to be set to the length of the body of the client certificate.

LOCALITY(ptr-ref)
Specifies a pointer reference to be set to the address of the locality from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

LOCALITYLEN(data-area)
Specifies a fullword binary data area to be set to the length of the locality from the client certificate.

ORGANIZATION(ptr-ref)
Specifies a pointer reference to be set to the address of the organization from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

ORGANIZATLEN(data-area)
Specifies a fullword binary data area to be set to the length of the organization from the client certificate.

ORGUNIT(ptr-ref)
Specifies a pointer reference to be set to the address of the organization unit from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

ORGUNITLEN(data-area)
Specifies a fullword binary data area to be set to the length of the organization unit from the client certificate.

OWNER
Indicates that the values returned by this command refer to the owner of the certificate.

SERIALNUM(ptr-ref)
Specifies a pointer reference to be set to the address of the serial number of the certificate assigned by the certificate issuer. The pointer reference is valid until the next CICS command or the end of task.

SERIALNUMLEN(data-area)
Specifies a fullword binary data area to be set to the length of the serial number.

STATE(ptr-ref)
Specifies a pointer reference to be set to the address of the state or province from the client certificate. The pointer reference is valid until the next CICS command or the end of task.

STATELEN(data-area)
Specifies a fullword binary data area to be set to the length of the state or province from the client certificate.

USERID(data-area)
Specifies an 8-byte field to be set to the user ID connected with the client certificate.

Conditions

16 INVREQ
occurs for the following conditions:

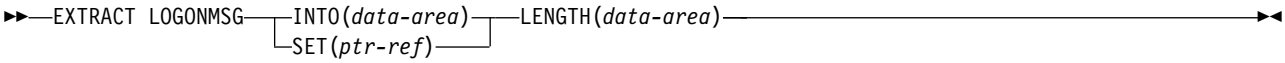
- The command is being issued in a non-CICS Web Interface application.
- The command is being issued for a non-HTTP request.
- If an error occurs retrieving the certificate data from CICS intermediate storage.

22 LENGERR
The string being extracted is longer than the length specified for one of the options.

EXTRACT LOGONMSG

Access z/OS Communications Server logon data.

EXTRACT LOGONMSG



Condition: NOTALLOC

Description

EXTRACT LOGONMSG accesses z/OS Communications Server logon data. This data may have been specified by the terminal operator at logon or in the ISSUE PASS command, for example. This data is only available if the system initialization parameter LGNMSG=YES is specified. The data can only be extracted once. It is possible to force the first transaction that runs on the terminal to be that which issues EXTRACT LOGONMSG by using the the system initialization parameter GMTRAN.

All the logon data is extracted and its length placed in the field specified by the LENGTH option. Because the LENGTH option cannot be used to limit the amount of data extracted, it is recommended that a field of 256 bytes is always used for this option.

If you use the SET option, the z/OS Communications Server logon data is not freed until the session terminates (CLSDST). If you use the INTO option, the z/OS Communications Server logon data is copied into user storage and then freed.

Options

INTO(*data-area***)**
specifies the receiving field for the data extracted.

LENGTH(*data-area***)**
specifies the length, as a halfword binary value, of the data extracted. If no data is available, LENGTH is set to zero.

SET(*ptr-ref***)**
specifies the pointer reference that is to be set to the address of the data extracted. The pointer reference, unless changed by other commands or statements, is valid until the next EXTRACT LOGONMSG command or the end of task.

If DATALOCATION(ANY) is associated with the application program, the address of the data can be above or below the 16MB line.

If DATALOCATION(BELOW) is associated with the application program, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

Conditions

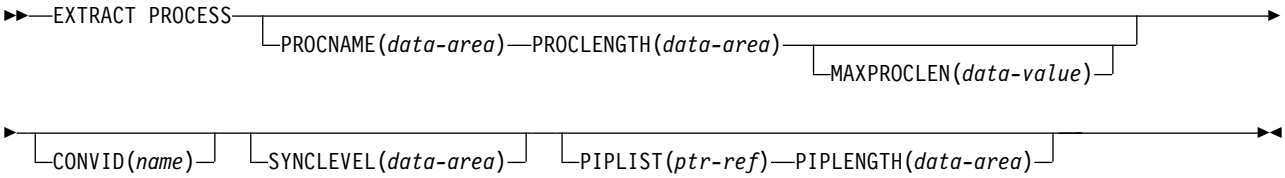
61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

EXTRACT PROCESS

Retrieve values from APPC conversation attach header.

EXTRACT PROCESS (APPC)



Conditions: INVREQ, LENGERR, NOTALLOC

Description

EXTRACT PROCESS lets an application program access conversation-related data, specified to CICS when the program is attached. The attach receiver does not have to execute an EXTRACT PROCESS command unless it requires this information.

The EXTRACT PROCESS command is valid only on an APPC conversation that is the principal facility for the task.

Options

CONVID(*name***)**
Identifies the conversation to which the command relates. The 4-character name identifies the token representing the principal session (EIBTRMID).

For compatibility with earlier releases, SESSION is accepted as a synonym for CONVID. New programs should use CONVID.

If CONVID and SESSION are both omitted, the principal facility for the task is used by default.

MAXPROCLEN(*data-value***)**
Specifies the buffer length of PROCNAME. If MAXPROCLEN is not specified, the buffer is assumed to have 32 bytes.

PIPLENGTH(*data-area***)**
Specifies a halfword binary data area in which the total length of the process initialization parameter (PIP) list is returned.

PIPLIST(*ptr-ref***)**
Specifies a pointer reference that is set to the address of a CICS-provided data area containing a PIP list. This list contains variable-length records in the same format as the list in the CONNECT PROCESS command. A returned value of zero means that no PIP data has been received by CICS.

PROCLENGTH(*data-area***)**
Specifies a halfword data area that is set by CICS to the length of the process name. If PROCNAME is specified, this option must be specified.

PROCNAME(*data-area***)**
Specifies the data area to receive the process name specified by the remote system that caused the task to start. The data area can be 1–64 bytes long. The process name is padded on the right with blanks if it is too short. The PROCNAME data area should not be shorter than the MAXPROCLEN value.

SYNCLEVEL(*data-area***)**
Specifies a halfword data area that is set by CICS to the SYNCLEVEL value. For further information about synchronization levels, see Synchronization.

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Also occurs (RESP2 not set) in any of the following situations:

- EXTRACT PROCESS has been used on a conversation other than APPC mapped (for example, LUTYPE6.1, APPC basic, or CPI Communications).

- EXTRACT PROCESS has been used on a conversation that was not started by input from the network, and whose session is not a principal facility.
- The command is issued against a CPI-Communications conversation.

Default action: terminate the task abnormally.

22 LENGERR
Occurs if the actual length of PROCNAME is greater than MAXPROCLEN, or greater than 32 bytes if MAXPROCLEN is not specified.

Default action: terminate the task abnormally.

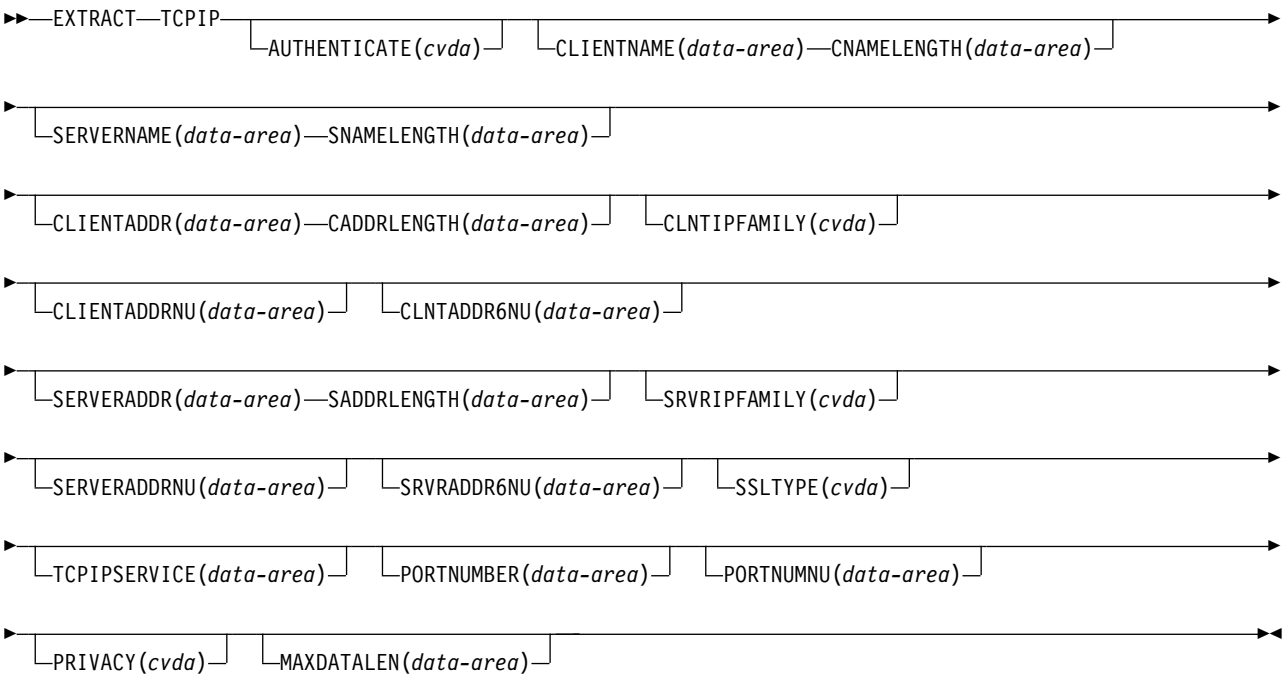
61 NOTALLOC
Occurs if the specified CONVID value specified does not relate to a conversation owned by the application.

Default action: terminate the task abnormally.

EXTRACT TCPIP

Obtain information about TCP/IP characteristics of the current transaction.

EXTRACT TCPIP



Conditions: INVREQ, LENGERR

This command is threadsafe.

Description

EXTRACT TCPIP provides information about the TCP/IP connection, and about security options specified in the TCPIPSERVICE definition.

Options

AUTHENTICATE*(cvda)*

Returns a CVDA indicating the authentication requested for the client using this transaction. Here are the values:

ASSERTED

AUTOAUTH

AUTOREGISTER

BASICAUTH

CERTIFICAUTH

NOAUTHENTIC

CADDRLENGTH*(data-area)*

Returns the length of the buffer supplied on the CLIENTADDR option, and is set to the length of the data returned to the application. If the CLIENTADDR is an IPv6 address, you must set the buffer length of CADDRLENGTH to at least 39 characters. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

CLIENTADDR*(data-area)*

Returns a buffer containing the IP address of the client. The IP address can be in IPv4 or IPv6 format. IPv4 addresses are returned as native IPv4 dotted decimal addresses; for example, `1.2.3.4` IPv6 addresses are returned as native IPv6 colon hexadecimal addresses; for example, `::a:b:c:d`

For information about IP addresses, see IP addresses.

CLIENTADDRNU*(data-area)*

Returns a fullword binary field containing the IPv4 address of the client in binary form. If the address is in IPv6 format, it is returned in the CLNTADDR6NU option and 0 is returned in CLIENTADDRNU.

CLIENTNAME*(data-area)*

Specifies a buffer to contain the name of the client as known by the Domain Name Server.

CLNTADDR6NU*(data-area)*

Returns a 16-byte field containing the IPv6 address of the client in binary form. This option is returned only if the option CLNTIPFAMILY has a value of `IPV6`. If the address is in IPv4 format, the address is returned in the CLNTADDRNU option and zeros are returned to CLNTADDR6NU.

CLNTIPFAMILY*(cvda)*

Returns the format of the IP address of the client. CVDA values are as follows:

IPV4 CLIENTADDR returns a dotted decimal IPv4 address and CLIENTADDRNU returns the IPv4 address in binary form.

IPV6 CLIENTADDR returns a colon hexadecimal IPv6 address and CLIENTADDR6NU returns the IPv6 address in binary form.

NOTAPPLIC
The source of the input has not been determined. `0.0.0.0` is returned.

CNAMELENGTH*(data-area)*

Specifies the length of the buffer supplied on the CLIENTNAME option, and is set to the actual length of the data returned to the application, or zero if the

name of the client is not known to the domain name server. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

MAXDATALEN*(data-area)*
Specifies a fullword binary field to contain the setting for the maximum length of data that can be received by CICS as an HTTP server.

PRIVACY*(cvda)*
Returns a CVDA indicating the level of SSL encryption used between the transaction and its client for an inbound request. CVDA values are as follows:

NOTSUPPORTED

REQUIRED

SUPPORTED

PORTNUMBER*(data-area)*
Specifies a 5-character field to contain the port number associated with this transaction in character form. This port received the incoming data that initiated this transaction.

PORTNUMNU*(data-area)*
Fullword field to contain the port number associated with this transaction in binary form. This port received the incoming data that initiated this transaction.

SADDRLENGTH*(data-area)*
Returns the length of the buffer supplied on the SERVERADDR option, and is set to the length of the data returned to the application. If SERVERADDR is an IPv6 address, you must set the buffer length of SADDRLENGTH to at least 39 characters. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

SERVERADDR*(data-area)*
Returns a buffer containing the IP address of the server. The IP address can be in IPv4 or IPv6 format. IPv4 addresses are returned as native IPv4 dotted decimal addresses, for example; `1.2.3.4`. IPv6 addresses are returned as native IPv6 colon hexadecimal addresses; for example, `::a:b:c:d`. If an error occurs, `0.0.0.0` is returned and the data is truncated.

SERVERADDRNU*(data-area)*
Returns a fullword binary field containing the IPv4 address of the server in binary form. If the address is IPv6 format, it is returned in the SRVRADDR6NU option and 0 is returned to SERVERADDRNU.

SERVERNAME*(data-area)*
Specifies a buffer to contain the name of the server as known by the Domain Name Server.

SNAMELENGTH*(data-area)*
Specifies the length of the buffer supplied on the SERVERNAME option, and is set to the length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

SRVRADDR6NU*(data-area)*
Returns a 16-byte field containing the IPv6 address of the server in binary form. This option is returned only if the option SRVIPFAMILY has a value of `IPV6`. If the address is in IPv4 format, the address is returned in the SERVERADDRNU option and zeros are returned in SRVRADDR6NU.

SRVRIPFAMILY*(cvda)*
Returns the format of the IP address of the server. CVDA values are as follows:

IPV4 SERVERADDR returns a dotted decimal IPv4 address and SERVERADDRNU returns the IPv4 address in binary form.

IPV6 SERVERADDR returns a colon hexadecimal IPv6 address and SERVERADDR6NU returns the IPv6 address in binary form.

NOTAPPLIC
The source of the input has not been determined. `0.0.0.0` is returned.

SSLTYPE*(cvda)*
Returns a CVDA indicating whether the Secure Sockets Layer (SSL) is being used to secure communications for this transaction. Here are the values:

SSL

NOSSL

CLIENTAUTH

ATTLSAWARE

TCPIPSERVICE*(data-area)*
An 8-byte field to contain the name of the TCPIPSERVICE associated with this transaction.

Conditions

16 INVREQ
RESP2 values:

2 An incorrect socket response was received.

5 The command was issued from a non-TCPIP application.

22 LENGERR
RESP2 values:

1 CLIENTADDR, SERVERADDR, CLIENTNAME, or SERVERNAME is specified, but the relevant length field is either not specified, or it is less than or equal to zero.

3 CLIENTADDR is too small to contain the string extracted.

4 SERVERADDR is too small to contain the string extracted.

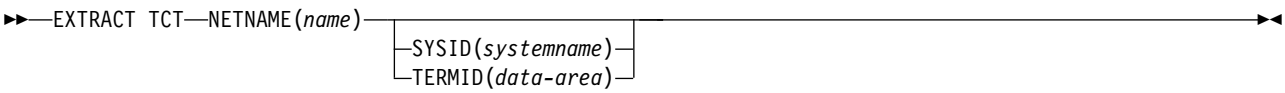
6 CLIENTNAME is too small to contain the string extracted.

7 SERVERNAME is too small to contain the string extracted.

EXTRACT TCT

Convert an 8-character name to a 4-character name on an LUTYPE6.1 logical unit.

EXTRACT TCT



Condition: INVREQ, NOTALLOC

Description

EXTRACT TCT converts the 8-character SNA network name for a logical unit into the corresponding 4-character name it is known by in the local CICS system.

Options

NETNAME(*name***)**
specifies the 8-character name of the logical unit in the SNA network.

SYSID(*systemname***)**
specifies the variable to be set to the equivalent local name of the system.

TERMID(*data-area***)**
specifies the variable to be set to the equivalent local name of the terminal.

Conditions

16 INVREQ
occurs if NETNAME is not valid.

Default action: terminate the task abnormally.

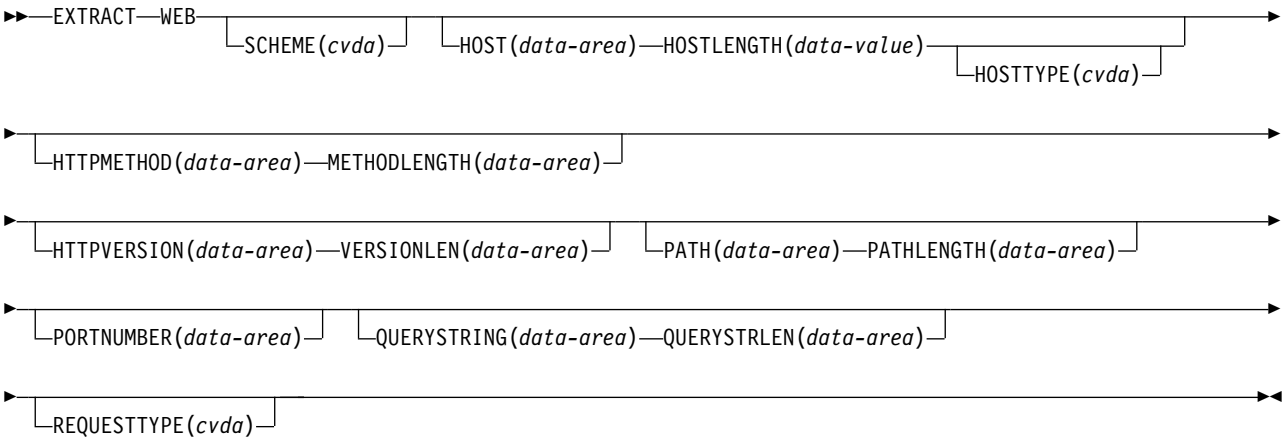
61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

EXTRACT WEB

Obtain information about an HTTP request that has been made to CICS as an HTTP server or about a connection between an Internet server and CICS as an HTTP client. This command is a synonym of WEB EXTRACT.

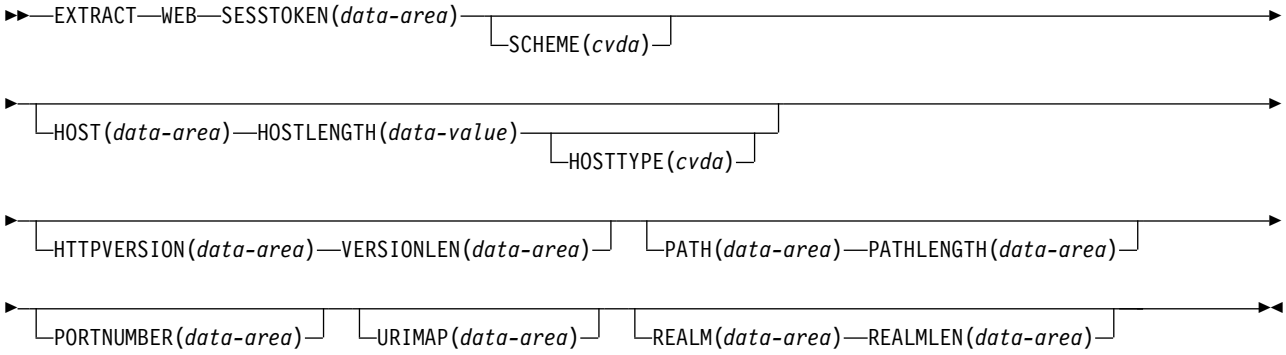
EXTRACT WEB (CICS as an HTTP server)



Conditions: INVREQ, LENGERR, NOTOPEN

This command is threadsafe.

EXTRACT WEB (CICS as an HTTP client)



Conditions: INVREQ, IOERR, LENGERR, NOTFND, NOTOPEN, TIMEDOUT

This command is threadsafe.

Description

For CICS as an HTTP server, WEB EXTRACT enables an application to obtain information about the most recent HTTP request that has been made to CICS by a Web client and assigned to the application for handling.

For CICS as an HTTP client, when the SESSTOKEN option is specified, the command enables an application to obtain information about a connection that it has opened with a server. The information returned to the application comprises global information about the connection, such as the host name of the server and its HTTP version. Information about specific requests made by the application, and responses made by the server, is not available using this command. The WEB RECEIVE command is used to receive information from a server response.

Options

HOST*(data-area)*

For CICS as an HTTP server, HOST specifies a buffer to contain the host component of the URL, as specified either in the Host header field for the request or in the request line (if an absolute URI was used for the request). The port number is presented separately using the PORTNUMBER option.

For CICS as an HTTP client, with the SESSTOKEN option, HOST specifies a buffer to contain the host name of the server in the connection identified by the SESSTOKEN option. The port number is presented separately using the PORTNUMBER option.

An IPv4 or IPv6 address can represent the host name. IPv4 addresses are returned as native IPv4 dotted decimal addresses; for example, `1.2.3.4`. IPv6 addresses are returned as native IPv6 colon hexadecimal addresses; for example, `::a:b:c:d`

For information on IP addresses, see IP addresses.

HOSTLENGTH*(data-area)*

Specifies the length of the buffer supplied on the HOST option, as a fullword binary variable, and is set to the length of the data returned to the application. 116 characters is an appropriate size to specify for this data area. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

HOSTTYPE*(cvda)*
Returns the address format of the HOST option. CVDA values are as follows:

HOSTNAME
The HOST option contains a character host name. The IP address that corresponds to the host name is looked up in the domain name server.

IPV4 The address is a dotted decimal IPv4 address.

IPV6 The address is a colon hexadecimal IPv6 address.

NOTAPPLIC
An incorrect host address was returned (`HOST=0.0.0.0`).

HTTPMETHOD*(data-area)*
For CICS as an HTTP server, this option specifies a buffer to contain the HTTP method string on the request line of the message.

This option is not relevant for CICS as an HTTP client.

HTTPVERSION*(data-area)*
For CICS as an HTTP server, this option specifies a buffer to contain the HTTP version for the Web client, as stated on its request.

For CICS as an HTTP client (with the SESSTOKEN option), this option specifies a buffer to contain the HTTP version of the server in the connection identified by the SESSTOKEN option. If CICS does not already know the HTTP version of the server, CICS makes a request to the server with the OPTIONS method to find out this information.

`1.1` indicates HTTP/1.1, and `1.0` indicates HTTP/1.0 or lower.

METHODLENGTH*(data-area)*
Specifies the length of the buffer supplied on the HTTPMETHOD option, as a fullword binary variable, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

PATH*(data-area)*
For CICS as an HTTP server, this option specifies a buffer to contain the path specified in the request line of the message.

For CICS as an HTTP client (with the SESSTOKEN option), this option specifies a buffer to contain the default path that applies to requests made using the connection. If a URIMAP definition was specified on the WEB OPEN command for the connection, the default path is the path specified in the URIMAP definition. Otherwise, the default path is a single forward slash.

PATHLENGTH*(data-area)*
Specifies the length of the buffer supplied on the PATH option, as a fullword binary variable, and is set to the length of the data returned to the application. 256 characters is an appropriate size to specify for this data-area. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

PORTNUMBER*(data-area)*
For CICS as an HTTP server, this option returns a data area containing the port number specified in the request line of the message.

For CICS as an HTTP client (with the SESSTOKEN option), this option returns a data containing the port number used to access the server in the connection specified by the SESSTOKEN option.

The value returned in the data area is a fullword binary value.

Well-known port numbers for a service are normally omitted from the URL. If the port number is not present in the URL, the command identifies and returns it based on the scheme. For HTTP, the well-known port number is 80, and, for HTTPS, the well-known port number is 443. If a port number is returned that is not the default for the scheme, you must specify the port number explicitly to gain access to the URL; for example, if you are using this information in a WEB OPEN command.

QUERYSTRING*(data-area)*
For CICS as an HTTP server, this option specifies a buffer to contain the query string on the request line of the message. The query string is the value or values encoded after the question mark (?) delimiting the end of the path. The query string is returned in its escaped form.

This option is not relevant for CICS as an HTTP client.

QUERYSTRLEN*(data-area)*
Specifies the length of the buffer supplied on the QUERY option, as a fullword binary variable, and is set to the length of the data returned to the application (the query string). 256 characters is an appropriate size to specify for this data area. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

REALM*(data-area)*
Specifies, for CICS as an HTTP client, the realm or security environment that contains the data that you are requesting. If you are issuing a command in response to an HTTP 401 message, REALM is the realm value in the most recently received WWW-Authenticate header.

REALMLEN*(data-area)*
Specifies, for CICS as an HTTP client, the buffer length supplied for the REALM option, as a fullword binary variable. If you are issuing a command in response to an HTTP 401 message, REALMLEN is the length of the realm name in the most recently received WWW-Authenticate header.

REQUESTTYPE*(cvda)*
For CICS as an HTTP server, this option specifies the type of request received. This option is not relevant for CICS as an HTTP client. CVDA values are as follows:

HTTPYES
Indicates an HTTP request.

HTTPNO
Indicates a non-HTTP request.

SCHEME*(cvda)*
For both CICS as an HTTP server, and CICS as an HTTP client (with the SESSTOKEN option), this option returns the scheme used for the connection between CICS and the Web client or server. CVDA values are as follows:

HTTP Is the HTTP protocol, without SSL.

HTTPS
Is the HTTPS protocol, which is HTTP with SSL.

SESSTOKEN*(data-value)*
For CICS as an HTTP client, this option is required. It specifies the session token, an 8-byte binary value that uniquely identifies a connection between CICS and a server. This value is returned by a WEB OPEN command for CICS as an HTTP client. Session tokens explains the use of the session token. For the command, information is returned about the specified connection.

This option is not relevant for CICS as an HTTP server.

URIMAP*(data-area)*
For CICS as an HTTP client (with the SESSTOKEN option), this option returns the 8-character name (in mixed case) of any URIMAP definition that was specified on the WEB OPEN command to open the connection specified by the SESSTOKEN option. The INQUIRE URIMAP command can be used to find information about the attributes of this URIMAP definition.

This option is not relevant for CICS as an HTTP server.

VERSIONLEN*(data-area)*
Specifies the length of the buffer supplied on the HTTPVERSION option, as a fullword binary variable, and is set to the length of the data returned to the application.

Conditions

16 INVREQ
RESP2 values:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request. This command is set only if one or more of HTTPMETHOD, HTTPVERSION, or PATH is specified and the request is a non-HTTP request).

41 The connection has closed. This is a WEB EXTRACT (Client) error only. The server may have timed out due to inactivity on this connection.

67 The content of the response does not conform to HTTP format. The error is generated because there is a syntax problem. This error is for WEB EXTRACT (Client) only.

71 A chunked transfer-coding error has occurred. This error is for WEB EXTRACT (Client) only.

144 One or more of the Web command parameters is invalid. This error is for WEB EXTRACT (Client) only.

17 IOERR
RESP2 values:

42 Socket error.

22 LENGERR
RESP2 values:

4 The method exceeds the length specified (METHODLENGTH option).

5 The PATHLENGTH option value was not greater than zero.

6 The HTTP version exceeds the length specified (VERSIONLEN option).

7 The VERSIONLEN option value was not greater than zero.

8 The query string exceeds the length specified (QUERYSTRLEN option).

21 The HOSTLENGTH option value was not greater than zero.

29 The host name exceeds the length specified (HOSTLENGTH option).

30 The path exceeds the length specified (PATHLENGTH option).

141 REALMLEN is not positive, or is not large enough to contain the realm value returned in the HTTP 401 response.

13 NOTFND
RESP2 values:

155 Request line information not found.

19 NOTOPEN
RESP2 values:

27 Session token not valid.

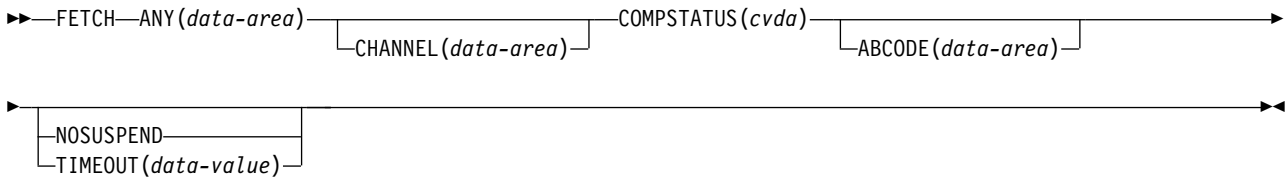
124 TIMEDOUT
RESP2 values:

62 Timeout on socket receive.

FETCH ANY

The **EXEC CICS FETCH ANY** command is used by a parent task to inquire on the status of any child task, and returns the status of any completed child task which has not yet been fetched.

FETCH ANY



Conditions: INVREQ, NOTFND, NOTFINISHED

Description

EXEC CICS FETCH ANY is used by a parent task to inquire on the status of any child task. **FETCH ANY** will return when any eligible child task has completed, or when there are no unfetched children remaining.

Options

ABCODE(*data-area***)**
Returns, if the child task terminated abnormally, the 4-character abend code. If an abend has not occurred, the variable is set to blanks.

ANY(*data-area***)**
Specifies a 16-character binary data area, into which CICS® will place the fetched child token. This will match the token returned by the **EXEC CICS RUN TRANSID** command.

CICS® will only return the token of a child task which has not yet been fetched.

CHANNEL(*data-area***)**
Returns the 16-character name of the reply channel of the child task identified by CHILD. The name is generated by CICS and is unique within the scope of the current link level. If the child task does not have a channel to return the response, then this field will contain blanks.

A child channel can only be fetched once, and the fetched channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

COMPSTATUS(*cvda***)**
Returns a CVDA value indicating the completion status of the child task. CVDA values are:

ABEND
The child task abended.

NORMAL
The child task completed normally.

SECERROR
The child task failed to attach due to a security error. ABCODE is blank.

NOSUSPEND
Specifies that the command will return immediately without waiting for any remaining child tasks to complete. If at least one of the unfetched children is completed, the command will return with a normal response.

TIMEOUT(*data-value***)**
Specifies a fullword binary data-value which is the maximum time in milliseconds that the command will wait for any remaining child tasks to complete before the command returns. The maximum value is 40800000 milliseconds. If TIMEOUT is not specified, or the specified value is zero, the command is not timed out.

Conditions

113 NOTFINISHED
RESP2 values:

52 NOSUSPEND was specified, but there were no completed children to fetch when the command was run.

53 No children have completed within the interval specified by the TIMEOUT option.

13 NOTFND
RESP2 values:

1 No unfetched children were found.

16 INVREQ
RESP2 values:

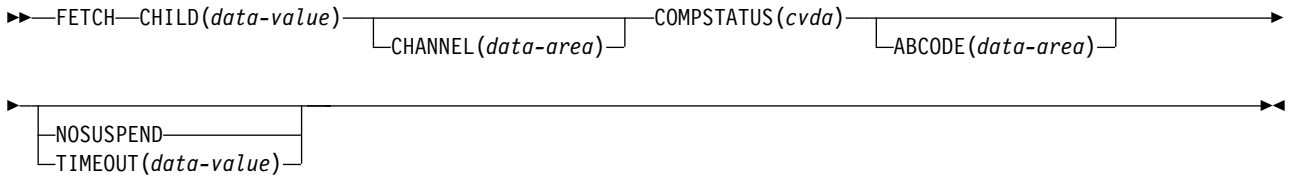
52 The parent has no children.

241 TIMEOUT value is not valid.

FETCH CHILD

The **EXEC CICS FETCH CHILD** command is used by a parent task to inquire on the status of a specific child task, and returns the status of the specified child task.

FETCH CHILD



Conditions: INVREQ, NOTFINISHED

Description

EXEC CICS FETCH CHILD is used by a parent task to inquire on the status of a specific child task, which is identified by the child token. **FETCH CHILD** will return when the child task is completed.

Options

ABCODE(*data-area***)**
Returns, if the child task terminated abnormally, the 4-character abend code. If an abend has not occurred, the variable is set to blanks.

CHANNEL(*data-area***)**
Returns the 16-character name of the reply channel of the child task identified by CHILD. The name is generated by CICS and is unique within the scope of the current link level. If the child task does not have a channel to return the response, then this field will contain blanks.

A child channel can only be fetched once, and the fetched channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

CHILD(*data-value***)**
Specifies the child token to fetch the response from. The child token will have been returned from the RUN TRANSID command.

COMPSTATUS(*cvda***)**
Returns a CVDA value indicating the completion status of the child task. CVDA values are:

ABEND
The child task abended.

NORMAL
The child task did not abend.

SECERROR
The child task failed to attach due to a security error. ABCODE is blank.

NOSUSPEND
Specifies that the command will return immediately without waiting for the child task to complete. If the child task is already completed when this command is issued, the command will return with a normal response.

TIMEOUT(*data-value***)**
Specifies a fullword binary data-value which is the maximum time in milliseconds that the command will wait for the requested child to complete

before the command returns. The maximum value is 4080000 milliseconds. If TIMEOUT is not specified, or the specified value is zero, the command is not timed out.

Conditions

113 NOTFINISHED
RESP2 values:

52 NOSUSPEND was specified, but the child is not yet complete.

53 The child has not completed within the interval specified by the TIMEOUT option.

16 INVREQ
RESP2 values:

50 The data that is specified in CHILD is not a valid child token, or the child task has already been freed by a **FREE CHILD** command.

51 The channel of the child has already been fetched.

241 TIMEOUT value is not valid.

FORCE TIMER

Force the early expiry of a BTS timer.

FORCE TIMER



Conditions: INVREQ, TIMERERR

Description

FORCE TIMER forces a BTS timer that has not yet expired to expire immediately. This causes the event associated with the timer to fire.

If the timer has already expired, the command has no effect.

The activity that owns the timer can be identified:

- Explicitly, by specifying either the ACQPROCESS or ACQACTIVITY option.
- Implicitly, by omitting the ACQPROCESS and ACQACTIVITY options. If these are omitted, the current activity is implied.

Options

ACQACTIVITY
specifies either of the following:

- If the program that issues the command has acquired a process, that the timer is owned by the root activity of that process.
- Otherwise, that the timer is owned by the activity that the program has acquired by means of an ACQUIRE ACTIVITYID command.

ACQPROCESS
specifies that the timer is owned by the process that the program that issues the command has acquired in the current unit of work.

TIMER(data-value)
specifies the name (1–16 characters) of the timer to be forced.

Conditions

16 INVREQ
RESP2 values:

1 The command was issued outside the scope of a currently-active activity.

16 The ACQPROCESS option was specified, but there is no acquired process.

17 The ACQACTIVITY option was specified, but there is no acquired activity.

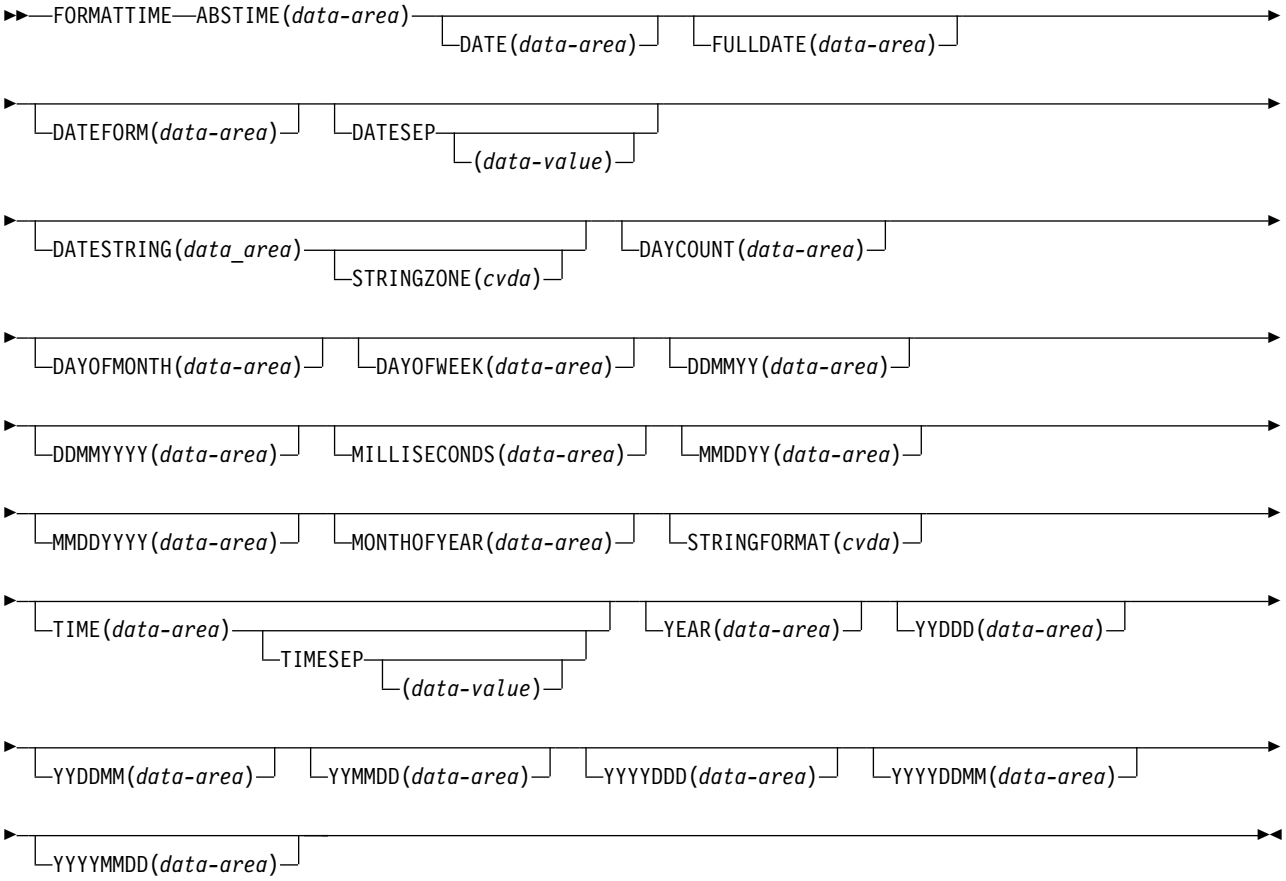
115 TIMERERR
RESP2 values:

13 The timer named on the TIMER option does not exist.

FORMATTIME

Transform absolute date and time into a specified format.

FORMATTIME



Condition: INVREQ

This command is threadsafe.

Description

FORMATTIME transforms the absolute date and time into any of a variety of formats. Normally, the ABSTIME argument is the value returned by an ASKTIME ABSTIME command.

To obtain an elapsed time in a particular format, the ABSTIME data value can be the difference between two values returned by ASKTIME, and options such as DAYCOUNT(d) and TIME(t) can be specified.

When you use the DATESTRING option to request an architected date and time stamp string that requires the clock time to be at UTC, CICS calculates the required timezone offset from your supplied ABSTIME value, which is in local time, and produces the date and time stamp string in UTC. All other values returned by the FORMATTIME command, such as the TIME value, are returned in local time. It is therefore normal for the FORMATTIME command to produce an architected date and time stamp string showing one date and time, and return other values showing another date and time, for the same supplied ABSTIME value.

Options

ABSTIME(*data-area***)**
Specifies the data area for the number of milliseconds since 00:00 on 1 January 1900, which is known as absolute time. The time is taken from the system time-of-day clock, adjusted for leap seconds and to apply the local timezone offset (including daylight saving time), truncated to the millisecond, and returned as a packed decimal of length 8 bytes. You can use FORMATTIME to change the data into other familiar formats.

The format of the parameter is:

```
COBOL:  PIC S9(15) COMP-3
C:      char data_ref[8];
PL/I:   FIXED DEC(15);
ASM:    PL8
```

DATE(*data-area***)**
Specifies the variable that is to receive the date in the format specified in the DATFORM system initialization parameter. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field. You normally use this option only when a date is needed for output purposes. Where a date is needed for analysis, request the date in explicit form; for example, using the MMDDYY option.

DATEFORM(*data-area***)**
Specifies the format of the installation-defined date. CICS returns YYMMDD, DDMMYY, or MMDDYY (six characters) according to the DATFORM system initialization parameter.

DATESEP(*data-value***)**
Specifies the character to be inserted as the separator between the year and the month and between the day and the month; or between the year and the day, if form YYDDD is specified.

If you omit this option, no separator is supplied. If you omit *data-value*, a slash (/) is assumed as the separator.

DATESTRING(*data-area*)
Specifies the 64-character user field where CICS returns the architected date and time stamp string in the format specified by the STRINGFORMAT option. If STRINGFORMAT is not specified, the default format provided is the RFC 1123 format (RFC1123). If you are using the DATESTRING option, first run the ASKTIME ABSTIME command to obtain a value for the ABSTIME option. If the value for the ABSTIME option is from any other source, the architected date and time stamp string that is returned by the FORMATTIME command might be incorrect.

DAYCOUNT(*data-area***)**
Returns the number of days since 1 January 1900 (day 1), as a fullword binary number. This function is useful if you need to compare the current date with a previous date that has, for example, been stored in a data set.

DAYOFMONTH(*data-area***)**
Returns the number of the day in the month as a fullword binary number.

DAYOFWEEK(*data-area***)**
Returns the relative day number of the week as a fullword binary number: Sunday=0, Saturday=6. This number can be converted to a textual form of day in any language.

DDMMYY(*data-area***)**
Specifies the 8-character user field where CICS returns the date, in day/month/year format; for example, 21/10/98. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field.

DDMMYYYY(*data-area***)**
Specifies the 10-character user field where CICS returns the date, in day/month/year format; for example 17/06/1995. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 10-character user field.

FULLDATE(data-area)
Specifies the 10-character user field where CICS returns the date, in the format specified in the DATFORM system initialization parameter, with the year expanded to four digits. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 10-character user field. You normally use this option only when a date is needed for output purposes. Where a date is needed for analysis, request the date in explicit form, for example, using the MMDDYYYY option.

MILLISECONDS(*data-area***)**
Returns the number of milliseconds in the current second specified by ABSTIME, as a binary integer in the range 0 - 999.

MMDDYY(*data-area***)**
Specifies the 8-character user field in which CICS returns the date, in month/day/year format; for example, 10/21/95. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field.

MMDDYYYY(*data-area***)**
Specifies the 10-character user field where CICS returns the date, in month/day/year format; for example 11/21/1995. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 10-character user field.

MONTHOFYEAR(*data-area***)**
data-area is set to the relative month number of the year as a fullword binary number (January=1, December=12). You can convert this number, in your application program, to the name of the month in any language.

STRINGFORMAT*(cvda)*
Specifies the format for the architected date and time stamp string returned in DATESTRING. The CVDA values are:

RFC1123
Specifies the RFC 1123 format, which is suitable for use in HTTP messages. This date and time stamp string contains the day, date, and 24-hour clock time, for example “Tue, 01 Apr 2003 10:01:02 +0000”. This format does not include milliseconds, and the number of seconds is truncated.

RFC3339
Specifies the RFC 3339 format, also known as the XML dateTime data type. This format is an implementation of a subset of the ISO 8601

standard. An example of a date and time stamp in this format is 2003-04-24T10:01:02+00:00. Date and time stamps in this format are in UTC (Coordinated Universal Time). This date and time stamp string contains the date and the 24-hour clock time. The time zone offset (-12:00 to +12:00) is indicated at the end of the date and time stamp.

The RFC 3339 specification allows the letter Z to be used for a zero offset (+00:00). A decimal fraction of a second in the 24-hour clock time is optional in the specification, and the FORMATTIME command does not include it. An example of a timestamp showing the decimal fraction of a second and the letter Z for a zero offset is 2003-04-01T10:01:02.498Z. If you want to add the decimal fraction of a second using your application, you can use the MILLISECONDS option to return the number of milliseconds that have also elapsed.

STRINGZONE*(cvda)*
Specifies the timezone in which the time stamp returned in DATESTRING is to be returned. The CVDA values are:

UTC DATESTRING is to be returned in UTC. This is the default setting.

LOCAL
DATESTRING is to be returned in LOCAL timezone.

TIME(*data-area***)**
data-area is set as an 8-character field to the current 24-hour clock time in the form hh:mm:ss, where the separator is specified by the TIMESEP option. The number of seconds is truncated. Use the MILLISECONDS option to return the number of milliseconds that have also elapsed.

TIMESEP(*data-value***)**
Specifies the character to be used as the separator in the returned time. If you omit this option, no separator is assumed and 6 bytes are returned in an 8-character field. If you omit *data-value*, a colon (:) is used as a separator.

YEAR(*data-area***)**
Specifies the full 4-figure number of the year as a fullword binary number; for example, 1995, 2001.

YYDDD(*data-area***)**
Specifies the 6-character user field where CICS returns the date, in year/day format; for example, 95/301. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 6-character user field.

YYDDMM(*data-area***)**
Specifies the 8-character user field where CICS returns the date, in year/day/month format; for example, 95/30/10. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field.

YYMMDD(*data-area***)**
Specifies the 8-character user field where CICS returns the date, in year/month/day format; for example, 95/10/21. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field.

YYYYDDD(*data-area***)**
Specifies the 8-character user field where CICS returns the date, in year/day format; for example 1995/200. A separator is present if requested by the

DATESEP option. If no separator is requested, the returned value is left-justified in the 8-character user field.

YYYYDDMM(*data-area***)**
Specifies the 10-character user field where CICS returns the date, in year/day/month format; for example 1995/21/06. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 10-character user field.

YYYYMMDD(*data-area***)**
Specifies the 10-character user field where CICS returns the date, in year/month/day format; for example 1995/06/21. A separator is present if requested by the DATESEP option. If no separator is requested, the returned value is left-justified in the 10-character user field.

Conditions

16 INVREQ
RESP2 values:

1 The ABSTIME value is less than zero or not in packed-decimal format.

2 Invalid CVDA value for the STRINGFORMAT option.

Default action: terminate the task abnormally.

Examples

The following example shows the effect of some of the options of the command. Let *utime* contain the value 003578979940458 in milliseconds.

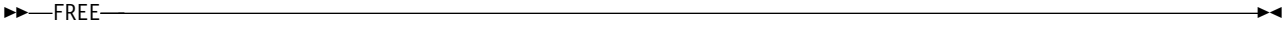
```
EXEC CICS ASKTIME ABSTIME(utime)
EXEC CICS FORMATTIME ABSTIME(utime)
          DATESEP('-') DDMMYY(date)
          TIME(time) TIMESEP(':')
```

This gives the values 05-31-13 for *date* and 08:05:40 for *time*.

FREE

Return a terminal or logical unit.

FREE



Condition: NOTALLOC

Description

FREE returns a terminal or logical unit when the transaction owning it no longer requires it. The principal facility is freed.

If you are running EDF, and the transaction frees the principal facility, EDF is terminated.

Conditions

61 NOTALLOC
occurs if the task is not associated with the terminal.

Default action: terminate the task abnormally.

FREE (APPC)

Return an APPC mapped session to CICS.

FREE (APPC)



Conditions: INVREQ, NOTALLOC

Description

FREE returns an APPC session to CICS when a transaction owning it no longer requires it. The session can then be allocated for use by other transactions.

If you omit CONVID, the principal facility is freed. Facilities not freed explicitly are freed by CICS when the task terminates.

If you are running EDF, and the transaction frees the principal facility, EDF is terminated.

Options

CONVID(*name*)
identifies the APPC mapped session to be freed. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

If this option is omitted, the principal facility is assumed.

STATE(*cvda*)
gets the state of the current conversation. The STATE option on a FREE command returns a cvda code of 00 if there is no longer an active conversation. The other output cvda values are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK

- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

also occurs (RESP2 not set) in any of the following situations:

- The CONVID value specified in the command relates to a basic (unmapped) APPC conversation.
- The CONVID value specified in the command relates to a CPI-Communications conversation.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the specified CONVID value does not relate to a conversation owned by the application.

Default action: terminate the task abnormally.

FREE (LUTYPE6.1)

Return LUTYPE6.1 sessions to CICS.

FREE (LUTYPE6.1)



Conditions: INVREQ, NOTALLOC

Description

FREE returns an LUTYPE6.1 session to CICS when a transaction that owns it no longer requires it. The session can then be allocated for use by other transactions.

If you omit both CONVID and SESSION, the principal facility is freed. Facilities not freed explicitly are freed by CICS when the task terminates.

If you are running EDF, and the transaction frees the principal facility, EDF is terminated.

Options

CONVID(*name*)
Identifies the LUTYPE6.1 session to be freed. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in

EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

SESSION(*name*)
Specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used.

Conditions

16 INVREQ
Occurs if the session specified in the command was allocated for a basic (unmapped) APPC conversation.

See also EIBRCODE in EIB fields.

Default action: terminate the task abnormally.

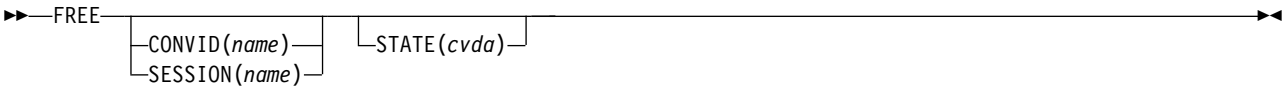
61 NOTALLOC
Occurs if the session specified in the command is not owned by the application.

Default action: terminate the task abnormally.

FREE (MRO)

Return MRO sessions to CICS.

FREE (MRO)



Conditions: INVREQ, NOTALLOC

Description

FREE returns an MRO session to CICS when a transaction that owns it no longer requires it. The session can then be allocated for use by other transactions.

If you omit both CONVID and SESSION, the principal facility is freed. Facilities not freed explicitly are freed by CICS when the task terminates.

If you are running EDF, and the transaction frees the principal facility, EDF is terminated.

Options

CONVID(*name*)
Identifies the MRO session to be freed. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

SESSION(*name*)
Specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used.

STATE(*cvda*)
Gets the state of the current conversation. The STATE on a FREE command returns a cvda code of 00 if there is no longer an active conversation. The other output cvda values are:

- ALLOCATED
- FREE
- PENDFREE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

16 INVREQ
Occurs in any one of the following situations:

- The session specified in the command was allocated for a basic (unmapped) APPC conversation
- The session is in the wrong state to be freed.

See also EIBRCODE in EIB fields.

Default action: terminate the task abnormally.

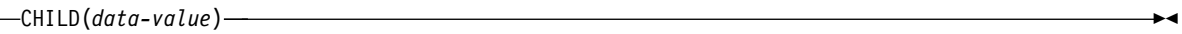
61 NOTALLOC
occurs if the session specified in the command is not owned by the application.

Default action: terminate the task abnormally.

FREE CHILD

The **EXEC CICS FREE CHILD** command frees a specified child token which has been previously allocated by an **EXEC CICS RUN TRANSID** command. If a parent task no longer requires the response of a child task, then using the **FREE CHILD** command will free the resources associated with that child task when it completes, rather than waiting for them to be fetched.

FREE CHILD



Conditions: INVREQ

Description

FREE CHILD is used by a parent task to free a child token allocated by a previous **RUN TRANSID** command. If the associated child task has already completed, then the memory object associated with the token is freed immediately, and CICS will also free any child channel (providing it has not been fetched using an **EXEC CICS FETCH** command).

If the child task has not completed, the memory object will be freed when the child completes, along with any child channel. After this command is issued, any subsequent attempt to reference the token with the **FETCH CHILD** command will fail with INVREQ, nor will the token be returned by any subsequent **FETCH ANY** commands.

CICS implicitly frees all child tokens created by **RUN TRANSID** when the parent task terminates.

Options

CHILD(*data-value***)**
Specifies the child token to be freed. The child token will have been returned from the **RUN TRANSID** command.

Conditions

16 INVREQ
RESP2 values:

50 The data that is specified in CHILD is not a valid child token, or the child task has already been freed by a **FREE CHILD** command.

FREEMAIN

Release main storage that was acquired by using a GETMAIN request.

FREEMAIN



Condition: INVREQ

This command is threadsafe.

Description

FREEMAIN releases the following storage:

- Main storage that was acquired by a GETMAIN command issued by the application.
- Main storage that was acquired by a LOAD command for a program, map, or table that is defined with RELOAD=YES.

If the task that acquired the storage or loaded the program does not release it, CICS releases the storage at task end, except for in the following situations:

- The GETMAIN command is specified the SHARED option. The storage remains allocated until another task issues a FREEMAIN or FREEMAIN64 request to release it.
- The program is defined with RELOAD=YES. The storage remains allocated until another task issues a FREEMAIN or FREEMAIN64 request to release it.
- The program is defined with RELOAD=NO but was loaded with the HOLD option. The program remains available until it is released by another task.

Note: In the first two situations listed, using FREEMAIN might create inter-transaction affinities that adversely affect the use of dynamic transaction routing. For more information about transaction affinities, see Affinity .

You can release CICS-key storage from a program only if it is being executed in CICS key. If the previously-acquired storage was obtained from CICS-key storage, and the program that issues the FREEMAIN request is in user-key, an INVREQ condition occurs with a RESP2 value of 2.

To release main storage that was acquired by using a GETMAIN64 request, use the FREEMAIN64 command. See FREEMAIN64.

Options

DATA(*data-area***)**
Specifies the data area of main storage to be released.

In assembler language, *data-area* must be a relocatable expression that is a data reference; in COBOL or C, it must be a data name; and in PL/I, it must be a data reference.

The length of storage released is the length that was obtained by the GETMAIN request and not necessarily the length of the data area.

DATAPOINTER(*ptr-value***)**
Specifies the address of the main storage to be released, as a pointer reference. This option is an alternative to the DATA option, and specifies the pointer reference that was returned by a GETMAIN command using the SET option.

The length of storage released is the length obtained by the GETMAIN request.

Conditions

16 INVREQ
RESP2 values:

1 The storage specified by the DATA or DATAPOINTER parameter is not storage acquired by a GETMAIN command.

2 The storage area specified by the DATA or DATAPOINTER parameter is in CICS-key storage, and the program issuing the FREEMAIN command is in user-key.

Default action: terminate the task abnormally.

Example: COBOL

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77  AREA-POINTER    USAGE IS POINTER.
LINKAGE SECTION.
  01  WORKAREA          PIC X(100).
PROCEDURE DIVISION.
  EXEC CICS GETMAIN SET(AREA-POINTER)
  LENGTH(100)
  END-EXEC.
      .
  SET ADDRESS OF WORKAREA TO AREA-POINTER.
      .
      .
  EXEC CICS FREEMAIN DATA(WORKAREA)
  END-EXEC.
  EXEC CICS RETURN
  END-EXEC.
```

Alternatively, the previous COBOL example could free the storage by using the following command:

```
EXEC CICS FREEMAIN DATAPOINTER(AREA-POINTER)
END-EXEC.
```

Example: C

```
#pragma XOPTS(CICS);
#define MAINSIZE 100;
main()
{
 char              *buffer;
 struct eib_record dfheiptr;
 EXEC CICS ADDRESS EIB(dfheiptr);
 EXEC CICS GETMAIN SET(buffer)
                   LENGTH(MAINSIZE);
 buffer[2] = 'a';
   .
   .
 EXEC CICS FREEMAIN DATA(buffer);
 EXEC CICS RETURN;
}
```

Example: PL/I

```
DCL AREA_PTR    POINTER,
    WORKAREA    CHAR(100) BASED(AREA_PTR);
  .
  .
  .
EXEC CICS GETMAIN SET(AREA_PTR) LENGTH(100);
  .
EXEC CICS FREEMAIN DATA(WORKAREA);
```

Example: Assembler

```
WORKAREA   DS   CL100
  .
  .
           EXEC CICS GETMAIN SET(9) LENGTH(100)
           USING  WORKAREA,9
           EXEC CICS FREEMAIN DATA(WORKAREA)
```

Alternatively, you can free storage using the DATAPOINTER option as shown in the following example:

```
WORKAREA  DS   CL100
  .
          EXEC CICS GETMAIN SET(9) LENGTH(100)
          USING  WORKAREA,9
  .
  .
          DROP   9
  .
          EXEC CICS FREEMAIN DATAPOINTER(9)
```

FREEMAIN64

Release storage that was acquired by using a GETMAIN or GETMAIN64 request. This command is for use only in non-Language Environment (LE) AMODE(64) assembler language application programs.

See Assembler language programming restrictions and requirements.

FREEMAIN64



Condition: INVREQ

This command is threadsafe.

Description

FREEMAIN64 releases the following storage:

- Main storage that was acquired by a GETMAIN or GETMAIN64 command issued by the application.
- Main storage that was acquired by a LOAD command for a program, map, or table that is defined with RELOAD=YES.

If the task that acquired the storage or loaded the program does not release it, CICS releases the storage at task end, except for in the following situations:

- The GETMAIN or GETMAIN64 command is specified the SHARED option. The storage remains allocated until another task issues a FREEMAIN or FREEMAIN64 request to release it.
- The program is defined with RELOAD=YES. The storage remains allocated until another task issues a FREEMAIN or FREEMAIN64 request to release it.
- The program is defined with RELOAD=NO but was loaded with the HOLD option. The program remains available until it is released by another task.

Note: In the first two situations listed, using FREEMAIN64 might create inter-transaction affinities that adversely affect the use of dynamic transaction routing. For more information about transaction affinities, see Affinity .

You can release CICS-key storage from a program only if it is being executed in CICS key. If the storage was obtained from CICS-key storage, and the program that issues the FREEMAIN64 request is in user-key, an INVREQ condition occurs with a RESP2 value of 2.

Options

DATA(*data-area64***)**
Specifies the data area of main storage to be released.

In assembler language, *data-area64* must be a relocatable expression that is a data reference.

The length of storage released is the length that was obtained by the original request and not necessarily the length of the data area.

DATAPOINTER(*ptr-value64***)**
Specifies the address of the main storage to be released, as a 64-bit pointer reference. This storage can be storage that was acquired by a previous GETMAIN or GETMAIN64 request. For example, a 64-bit pointer reference to an area of 31-bit storage can be specified.

The length of storage released is the length that was obtained by the original request.

Conditions

16 INVREQ
RESP2 values:

1 The storage specified by the DATA or DATAPOINTER parameter is not storage acquired by a GETMAIN or GETMAIN64 command.

2 The storage area specified by the DATA or DATAPOINTER parameter is in CICS-key storage, and the program issuing the FREEMAIN64 command is in user-key.

Examples: Assembler

The following example releases storage by using the DATA option.

```
WORKAREA   DS   CL100
  .
  .
           EXEC CICS GETMAIN64 SET(9) FLENGTH(1048576)
           USING  WORKAREA,9
           EXEC CICS FREEMAIN64 DATA(WORKAREA)
```

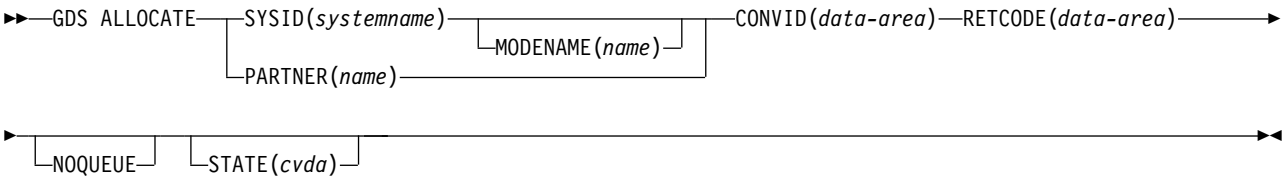
The following example releases storage by using the DATAPOINTER option.

```
WORKAREA  DS   CL100
  .
         EXEC CICS GETMAIN SET(9) LENGTH(100)
         USING  WORKAREA,9
  .
  .
         DROP   9
  .
         EXEC CICS FREEMAIN64 DATAPOINTER(9)
```

GDS ALLOCATE

Acquire a session to a remote system for use by APPC basic conversation (assembler-language and C programs only).

GDS ALLOCATE (APPC basic)



Description

GDS ALLOCATE acquires a session to a remote system.

The return code is given in RETCODE (see Table 1 on page 211). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONVID(*data-area***)**
specifies the 4-character application data area that is to contain the token returned by an ALLOCATE command to identify the allocated conversation. This token is required in subsequent GDS commands issued on the conversation.

MODENAME(*name***)**
specifies the name of the mode group from which the session is to be acquired. If you specify SYSID and omit MODENAME, CICS selects a modename from those defined for the system.

NOQUEUE
specifies that the request to allocate a session is not to be queued when a suitable APPC session cannot be acquired immediately. A session is acquired immediately only if it is a bound contention winner that is not already allocated to another conversation.

The return code in RETCODE indicates whether or not a session has been acquired.

If the NOQUEUE option is not used, a delay may occur before control is passed back to the application program. A delay can occur for any of the following reasons:

- All sessions for the specified SYSID and MODENAME are in use.
- The CICS allocation algorithm has selected a session that is not currently bound (in which case, CICS has to bind).
- The CICS allocation algorithm has selected a contention loser (in which case, CICS has to bid).

If there is a delay, the program waits until the session has been acquired.

PARTNER(*name***)**
specifies the name (eight characters) of a set of definitions that include the names of a remote LU (NETNAME) and a communication profile to be used on the allocated session. For APPC basic conversations, the only relevant attribute set by the profile is MODENAME.

If you use this option as an alternative to SYSID and MODENAME, CICS uses the NETNAME and MODENAME from the PARTNER definition.

RETCODE(*data-area***)**
specifies the 6-byte application data area into which return code information (shown in Table 1) is to be moved.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

SYSID(*systemname***)**
specifies the remote system to which an APPC session is to be allocated. The name, which is 1–4 characters, identifies an entry (defined as an APPC connection) in the CICS terminal control table.

Table 1. GDS ALLOCATE return codes

RETCODE (hexadecimal)	**Description**
01 0C 00	SYSID is unrecognized.
01 0C 04	SYSID is not an LUTYPE6.2 connection name.
01 04 04	NOQUEUE is specified but no bound connection-winner sessions are available.

Table 1. GDS ALLOCATE return codes (continued)

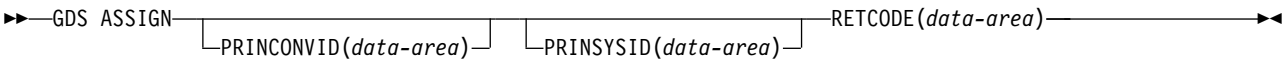
RETCODE (hexadecimal)	Description
01 04 08	MODENAME is not known.
01 04 0C	The MODENAME value is SNASVCMG which is restricted to use by CICS.
01 04 0C	z/OS Communications Server has no class of service (COS) table for the MODENAME value.
01 04 10	The task was canceled during queuing of the command.
01 04 14	All modegroups are closed.
01 04 14	The requested modegroup is closed.
01 04 18	The requested modegroup is draining (closing down).
01 08 00	All sessions in the requested modegroup are unusable.
01 08 00	The connection is in quiesce state.
01 08 00	The connection is out of service.
01 08 00	The connection is not acquired.
01 08 00	The requested modegroup's local max (maximum permitted number of sessions) is 0.
01 08 00	The VTAM® ACB is closed.
01 0C 14	The NETNAME specified in the PARTNER definition is not known.
02 0C 00	PARTNER is not known.
06 00 00	The PROFILE specified in the PARTNER definition is not known.

Note: VTAM is now the z/OS Communications Server.

GDS ASSIGN

Get the identifier of the principal facility in use by APPC basic conversation (assembler-language and C programs only).

GDS ASSIGN (APPC basic)



Description

GDS ASSIGN gets the identifier of the principal facility.

The return code is given in RETCODE (see Table 2 on page 213). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

PRINCONVID(*data-area*)
specifies a 4-byte data area in which the conversation token (CONVID) of the principal facility is to be returned.

PRINSYSID(*data-area*)
specifies a 4-byte data area in which the SYSID of the principal facility is to be returned.

RETCODE(*data-area*)
specifies the 6-byte application data area into which return code information (shown in Table 2) is to be moved.

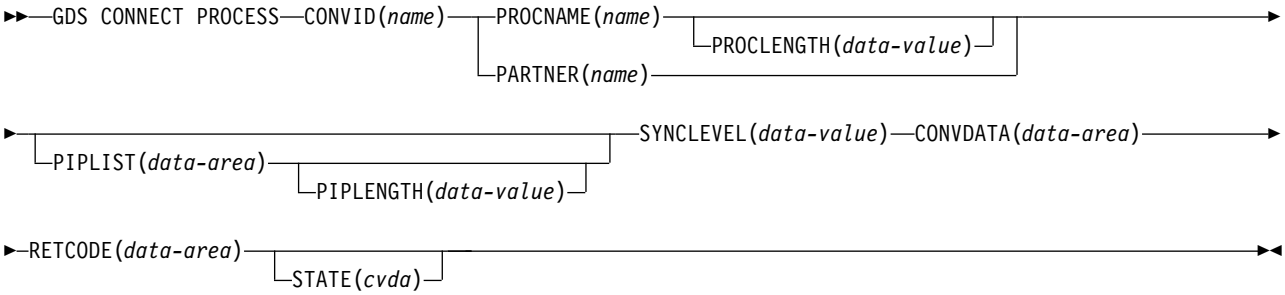
Table 2. GDS ASSIGN return codes

RETCODE (hexadecimal)	Description
03 00	Principal facility is not APPC.
03 04	Principal facility is not basic.
04	No terminal principal facility exists.

GDS CONNECT PROCESS

Initiate an APPC basic conversation (assembler-language and C programs only).

GDS CONNECT PROCESS (APPC basic)



Description

EXEC CICS conditions are never raised on GDS commands.

The return code is given in RETCODE (see Table 3 on page 215). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

GDS CONNECT PROCESS allows the application program to specify a partner application that is to run in the remote system.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONVDATA(*data-area*)
specifies the 24-byte application data area into which conversation-related

information is to be moved. A description of the format of the data area is given in the discussion of CONVDATA fields in the Testing indicators.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed GDS ALLOCATE command, or the token representing the principal session (returned by a previously executed GDS ASSIGN command).

PARTNER(*name***)**
specifies the name (8 characters) of a set of definitions that includes the name (or extended name) of a remote partner transaction (TPNAME or XTPNAME). You can use this option as an alternative to PROCNAME and PROCLENGTH.

PIPLENGTH(*data-value***)**
specifies the total length of the process initialization parameter (PIP) list specified on a CONNECT PROCESS command.

PIPLIST(*data-area***)**
specifies the PIP data that is to be sent to the remote process.

PROCLENGTH(*data-value***)**
specifies the length (as a halfword binary value in the range 1–64) of the target process name.

PROCNAME(*name***)**
specifies the name of the remote application. The APPC architecture allows names of lengths (1–64 bytes), but leaves each product free to set its own maximum. If the remote system is CICS, you can use the standard 4-character transaction ID. You can also use the TPNAME value in the transaction definition.

RETCODE(*data-area***)**
specifies the 6-byte application data area into which return code information (shown in Table 3 on page 215) is to be moved.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

SYNCLEVEL(*data-value***)**
specifies the synchronization level (halfword binary value) intended for the current conversation. The possible values are:

- 0 None

- 1 Confirm
- 2 Syncpoint

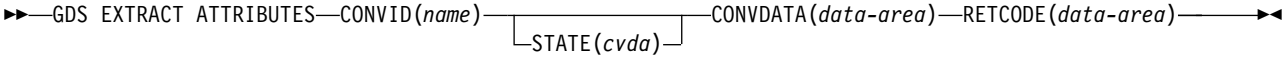
Table 3. GDS CONNECT PROCESS return codes

RETCODE (hexadecimal)	Description
02 0C 00	PARTNER is not known.
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 04	CONVID is for a conversation that is not basic.
03 0C	The SYNCLEVEL option specifies a value other than 0, 1, or 2.
03 0C	The SYNCLEVEL option requested either 1 or 2, but it was unavailable.
03 08	A state check occurred.
04	CONVID is for a session that is not allocated to the task, or that is a relay link.
05 00 00 00 00 20	PROCLENGTH is outside the range 1–64.
05 00 00 00 7F FF	The PIPLENGTH value is outside the range 4–763.
05 00 00 00 7F FF	The 2-byte length field (LL) for one of the PIPs is less than 4.
05 00 00 00 7F FF	The total of the LLs in PIP data is greater than the PIPLENGTH value.

GDS EXTRACT ATTRIBUTES

Access state information on an APPC basic conversation (assembler-language and C programs only).

GDS EXTRACT ATTRIBUTES (APPC basic)



Description

GDS EXTRACT ATTRIBUTES accesses state information about an APPC basic conversation.

The return code is given in RETCODE (see Table 4 on page 216). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed GDS

ALLOCATE command, or the token representing the principal session (returned by a previously executed GDS ASSIGN command).

CONVDATA(*data-area***)**
specifies the 24-byte application data area into which conversation-related information is to be moved. A description of the format of the data area is given in the discussion of CONVDATA fields in the Testing indicators.

RETCODE(*data-area***)**
specifies the 6-byte application data area into which return code information (shown in Table 4) is to be moved.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

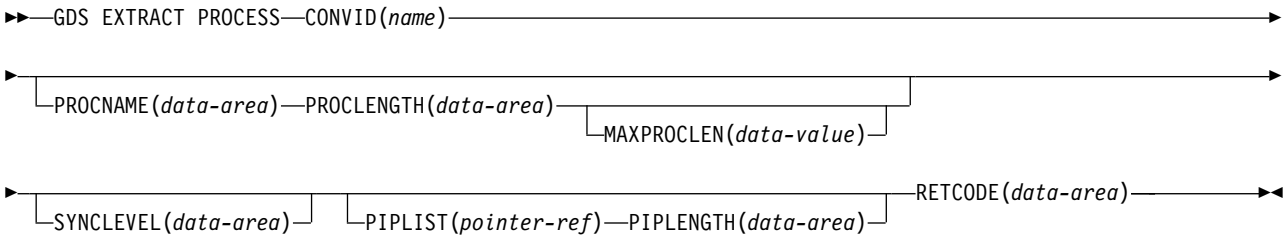
Table 4. GDS EXTRACT ATTRIBUTES return codes

RETCODE (hexadecimal)	Description
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 01	INVREQ for a DPL server program.
03 04	CONVID is for a conversation that is not basic.
04	CONVID is for a session that is not allocated to the task, or that is a relay link.

GDS EXTRACT PROCESS

Retrieve values from an APPC basic conversation (assembler-language and C programs only).

GDS EXTRACT PROCESS (APPC basic)



Description

GDS EXTRACT PROCESS retrieves values from an APPC basic conversation. The data retrieved is valid only when the command is issued against an APPC basic principal facility.

The return code is given in RETCODE (see Table 5 on page 218). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONVID(*name***)**
identifies the conversation the command relates to. The 4-character name identifies the token representing the principal session (returned by a previously executed GDS ASSIGN command).

MAXPROCLEN(*data-value***)**
specifies the length (1–64 characters) of the PROCNAME data area. If MAXPROCLEN is not specified, the buffer is assumed to have 32 bytes.

PIPLENGTH(*data-area***)**
specifies a halfword binary data area that is to receive the length of the PIPLIST received by a GDS EXTRACT PROCESS command.

PIPLIST(*pointer-ref***)**
specifies the pointer reference that is to be set to the address of the PIPLIST received by a GDS EXTRACT PROCESS command. A zero setting indicates that no PIPLIST was received.

PROCLENGTH(*data-area***)**
specifies a halfword binary data area that is set to the actual length of the process name.

PROCNAME(*data-area***)**
specifies the application target data area (1–64 bytes) into which the process name, specified in the APPC attach function management header, is to be moved. The area is padded with blanks, if necessary.

RETCODE(*data-area***)**
specifies the 6-byte application data area into which return code information (shown in Table 5 on page 218) is to be moved.

SYNCLEVEL(*data-area***)**
specifies a halfword binary data area that is set to indicate the synchronization level in effect for the current conversation. The possible values are:

- 0 None
- 1 Confirm
- 2 Syncpoint

Table 5. GDS EXTRACT PROCESS return codes

RETCODE (hexadecimal)	Description
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 00	CONVID is for a session that is not the principal facility.
03 00	Principal facility was not started by terminal data.
03 04	CONVID is for a conversation that is not basic.
04	CONVID is for a session that is not allocated to the task, or that is a relay link.
05 00 00 00 00 20	PROCLENGTH value returned is greater than MAXPROCLEN value.

GDS FREE

Return an APPC session to CICS (assembler-language and C programs only).

GDS FREE (APPC basic)



Description

GDS FREE returns the session to CICS. The issue of this command is valid only when the conversation is finished, that is, the conversation state is FREE.

The return code is given in RETCODE (see Table 6 on page 219). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONVDATA(*data-area***)**
specifies the 24-byte application data area into which conversation-related information is to be moved. A description of the format of the data area is given in the discussion of CONVDATA fields in the Testing indicators.

CONVID(*name*)
identifies the conversation to be freed. The 4-character name identifies either the token returned by a previously executed GDS ALLOCATE command, or the token representing the principal session (returned by a previously executed GDS ASSIGN command).

RETCODE(*data-area*)
specifies the 6-byte application data area into which return code information (shown in Table 6) is to be moved.

STATE(*cvda*)
gets the state of the current conversation. The STATE on a FREE command returns a cvda code of 00 if there is no longer an active conversation. The other output cvda values are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

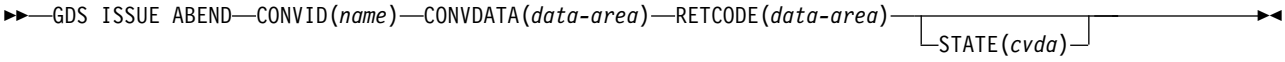
Table 6. GDS FREE return codes

RETCODE (hexadecimal)	Description
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 04	CONVID is for a conversation that is not basic.
03 08	A state check has occurred.
04	CONVID is for a session that is not allocated to the task, or that is a relay link.

GDS ISSUE ABEND

Terminate APPC basic conversation abnormally (assembler-language and C programs only).

GDS ISSUE ABEND (APPC basic)



Description

GDS ISSUE ABEND causes an APPC basic conversation to end immediately, regardless of the conversation state. The partner transaction is informed.

The return code is given in RETCODE (see Table 7). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONVDATA(*data-area***)**
specifies the 24-byte application data area into which conversation-related information is to be moved. A description of the format of the data area is given in the discussion of CONVDATA fields in the Testing indicators.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed GDS ALLOCATE command, or the token representing the principal session (returned by a previously executed GDS ASSIGN command).

RETCODE(*data-area***)**
specifies the 6-byte application data area into which return code information (shown in Table 7) is to be moved.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Table 7. GDS ISSUE ABEND return codes

RETCODE (hexadecimal)	Description
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 04	CONVID is for a conversation that is not basic.
03 08	A state check has occurred.

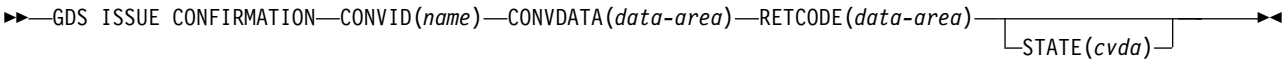
Table 7. GDS ISSUE ABEND return codes (continued)

RETCODE (hexadecimal)	Description
04	CONVID is for a session that is not allocated to the task, or that is a relay link.

GDS ISSUE CONFIRMATION

Issue a synchronization request on an APPC basic conversation (assembler-language and C programs only).

GDS ISSUE CONFIRMATION (APPC basic)



Description

GDS ISSUE CONFIRMATION issues a synchronization request in response to a GDS SEND CONFIRM issued by a partner transaction.

The return code is given in RETCODE (see Table 8 on page 222). For a list of return code values, see the Return codes for APPC basic conversationsReturn codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONVDATA(*data-area***)**
specifies the 24-byte application data area into which conversation-related information is to be moved. A description of the format of the data area is given in the discussion of CONVDATA fields in the Testing indicators.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed GDS ALLOCATE command, or the token representing the principal session (returned by a previously executed GDS ASSIGN command).

RETCODE(*data-area***)**
specifies the 6-byte application data area into which return code information (shown in Table 8 on page 222) is to be moved.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND

- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

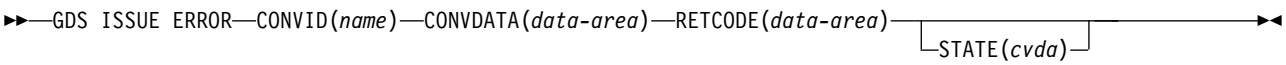
Table 8. GDS ISSUE CONFIRMATION return codes

RETCODE (hexadecimal)	**Description**
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 04	CONVID is for a conversation that is not basic.
03 08	A state check has occurred.
03 14	The command was issued for a sync level 0 conversation.
04	CONVID is for a session that is not allocated to the task, or that is a relay link.

GDS ISSUE ERROR

Inform an APPC basic conversation partner of an error (assembler-language and C programs only).

GDS ISSUE ERROR (APPC basic)



Description

GDS ISSUE ERROR informs the conversation partner that there is an error.

The return code is given in RETCODE, see below. For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONVDATA(*data-area***)**
specifies the 24-byte application data area into which conversation-related information is to be moved. A description of the format of the data area is given in the discussion of CONVDATA fields in the Testing indicators.

CONVID(*name*)
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed GDS ALLOCATE command, or the token representing the principal session (returned by a previously executed GDS ASSIGN command).

RETCODE(*data-area*)
specifies the 6-byte application data area into which return code information (shown in Table 9) is to be moved.

STATE(*cvda*)
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

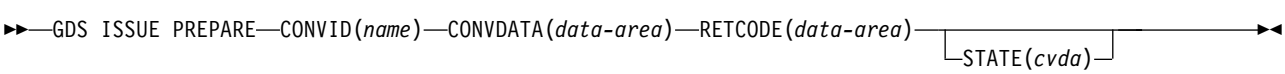
Table 9. GDS ISSUE ERROR return codes

RETCODE (hexadecimal)	Description
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 04	CONVID is for a conversation that is not basic.
03 08	A state check has occurred.
04	CONVID is for a session that is not allocated to task, or that is a relay link.

GDS ISSUE PREPARE

Issue the first flow of a syncpoint request on an APPC basic conversation (assembler-language and C programs only).

GDS ISSUE PREPARE (APPC basic)



Description

GDS ISSUE PREPARE issues the first flow of a syncpoint request.

The return code is given in RETCODE (see Table 10). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONVDATA(*data-area***)**
specifies the 24-byte application data area into which conversation-related information is to be moved. A description of the format of the data area is given in the discussion of CONVDATA fields in the Testing indicators.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed GDS ALLOCATE command, or the token representing the principal session (returned by a previously executed GDS ASSIGN command).

RETCODE(*data-area***)**
specifies the 6-byte application data area into which return code information (shown in Table 10) is to be moved.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Table 10. GDS ISSUE PREPARE return codes

RETCODE (hexadecimal)	Description
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 04	CONVID is for a conversation that is not basic.
03 0C	The command was issued on a conversation that is not sync-level 2.

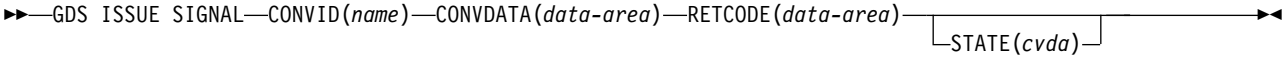
Table 10. GDS ISSUE PREPARE return codes (continued)

RETCODE (hexadecimal)	Description
03 24	A state error occurred.
04	CONVID is for a session that is not allocated to task, or that is a relay link.

GDS ISSUE SIGNAL

Request a change of direction from the sending transaction APPC basic conversation (assembler-language and C programs only).

GDS ISSUE SIGNAL (APPC basic)



Description

GDS ISSUE SIGNAL requests a change of direction.

The return code is given in RETCODE (see Table 11 on page 226). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONVDATA(*data-area***)**
specifies the 24-byte application data area into which conversation-related information is to be moved. A description of the format of the data area is given in the discussion of CONVDATA fields in the Testing indicators.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed GDS ALLOCATE command, or the token representing the principal session (returned by a previously executed GDS ASSIGN command).

RETCODE(*data-area***)**
specifies the 6-byte application data area into which return code information (shown in Table 11 on page 226) is to be moved.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE

- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

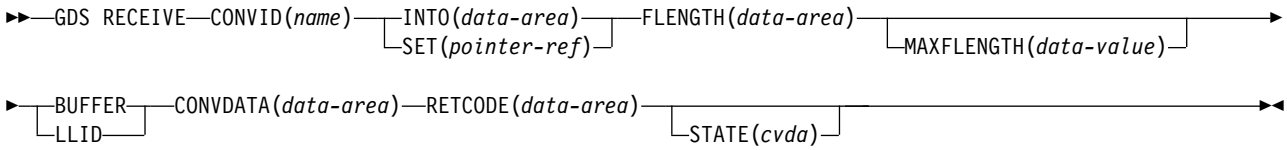
Table 11. GDS ISSUE SIGNAL return codes

RETCODE (hexadecimal)	Description
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 04	CONVID is for a conversation that is not basic.
03 08	A state check has occurred.
04	CONVID is for a session that is not allocated to task, or that is a relay link.

GDS RECEIVE

Receive data on an APPC basic conversation (assembler-language and C programs only).

GDS RECEIVE (APPC basic)



Description

GDS RECEIVE receives data and indicators from a partner transaction.

The return code is given in RETCODE (see Table 12 on page 228). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

BUFFER
specifies that the length of the data passed to the application program in response to the RECEIVE command is to be restricted only by the length specified in the MAXFLENGTH option, and is not to be affected by GDS structured field boundaries. Control is returned to the application program

when this length has been received, or when a synchronization request, change-direction, or end-bracket is received.

CONVDATA(*data-area***)**
specifies the 24-byte application data area into which conversation-related information is to be moved. A description of the format of the data area is given in the discussion of CONVDATA fields in the Testing indicators.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed GDS ALLOCATE command, or the token representing the principal session (returned by a previously executed GDS ASSIGN command).

FLENGTH(*data-area***)**
specifies a fullword binary data area that is set to the length of the data made available to the application program.

INTO(*data-area***)**
specifies the application target data area into which data is to be received from the application program connected to the other end of the current conversation. The length of this area must not be less than the value specified in the MAXFLENGTH option.

LLID
specifies that the delimiter to be used by CICS to terminate the passing of data to the application program is the end of a GDS structured field, if this occurs before the MAXFLENGTH limit is reached.

MAXFLENGTH(*data-value***)**
specifies, as a fullword binary value, either the length of the target data area specified in the INTO option, or the maximum length of data to be addressed by the pointer reference specified in the SET option. The length must not exceed 32 767 bytes. CICS does not receive more data than the MAXFLENGTH value allows.

RETCODE(*data-area***)**
specifies the 6-byte application data area into which return code information (shown in Table 12 on page 228) is to be moved.

SET(*pointer-ref***)**
specifies the pointer reference to be set to the address of data received from the application program connected to the other end of the current conversation. The pointer reference, unless changed by other commands or statements, is valid until the next RECEIVE (GDS or APPC) command, or the end of the task.

If DATALOCATION(ANY) is associated with the application program, the address of the data can be above or below the 16MB line.

If DATALOCATION(BELOW) is associated with the application program, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED

- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

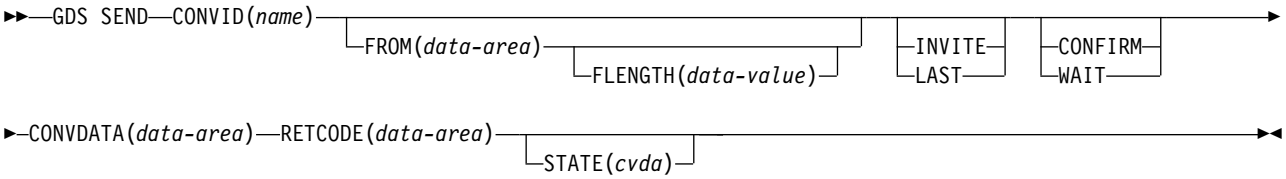
Table 12. GDS RECEIVE return codes

RETCODE (hexadecimal)	Description
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 04	CONVID is for a conversation that is not basic.
03 08	A state check occurred.
04	CONVID is for a session that is not allocated to the task, or that is a relay link.
05 00 00 00 7F FF	MAXFLENGTH is outside the range 0 through 32 767.

GDS SEND

Send data on an APPC basic conversation (assembler-language and C programs only).

GDS SEND (APPC basic)



Description

GDS SEND sends data.

The return code is given in RETCODE (see Table 13 on page 230). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONFIRM
allows an application working at synchronization level 1 or 2 to synchronize its processing with that of a process in a remote system. The actions taken to synchronize processing are defined by the application programs involved. The CONFIRM option causes RQD2 to be added to the data already sent, and forces a WAIT. On receipt of the indicator, the remote process takes the agreed actions and then sends a response. When the WAIT completes, CDBERR is set to X'00' if the appropriate response has been received.

CONVDATA(*data-area***)**
specifies the 24-byte application data area into which conversation-related information is to be moved. A description of the format of the data area is given in the discussion of CONVDATA fields in the Testing indicators.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed GDS ALLOCATE command, or the token representing the principal session (returned by a previously executed GDS ASSIGN command).

FLENGTH(*data-value***)**
specifies the length (as a fullword binary value in the range 1–32 767) of the data specified in the FROM option.

FROM(*data-area***)**
specifies the data that is to be sent.

INVITE
allows an application program to add a change-direction indicator to data already sent to a process in a connected APPC system. Control data is not transmitted by CICS until the subsequent execution of a WAIT or a SYNCPOINT command, unless CONFIRM or WAIT is also coded on the GDS SEND INVITE command.

LAST
allows an application program to add CEB to data already sent to a process in a connected APPC system. CEB is not transmitted by CICS until the subsequent execution of a WAIT or a SYNCPOINT command, unless CONFIRM or WAIT is also coded on the GDS SEND LAST command. Note that if one of these commands fails because of a conversation-related error, the conversation remains in bracket. In such a case, the application program should execute a GDS RECEIVE command. However, GDS SEND LAST WAIT (with no data) always causes the conversation to be deallocated.

RETCODE(*data-area***)**
specifies the 6-byte application data area into which return code information (shown in Table 13 on page 230) is to be moved.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE

- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

WAIT
ensures that all data and indicators so far sent on a conversation are erased from the partner transaction.

If the WAIT option is not used, data from successive SEND commands is accumulated by CICS, together with any indicators, in an internal buffer. If the buffer becomes full, most of the accumulated data is transmitted to the remote system, but the accumulated indicators are not. Transmission of the accumulated data plus the indicators is forced by the WAIT or CONFIRM options of the GDS SEND command, or by a GDS WAIT command.

Table 13. GDS SEND return codes

RETCODE (hexadecimal)	Description
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 04	CONVID is for a conversation that is not basic.
03 08	A state check has occurred.
03 14	The CONFIRM option has been used on a sync level 0 conversation.
03 10	LL error (incorrect or incomplete).
04	CONVID is for a session that is not allocated to the task, or that is a relay link.
05 00 00 00 7F FF	The FLENGTH value is outside the range 0 through 32 767.

GDS WAIT

Ensure accumulated data transmitted on an APPC conversation (assembler-language and C programs only).

GDS WAIT (APPC basic)



Description

GDS WAIT ensures that the accumulated data has been sent.

The return code is given in RETCODE (see Table 14). For a list of return code values, see the Return codes for APPC basic conversations. EXEC CICS conditions are never raised on GDS commands.

Options

In general, the arguments can be replaced by absolute or relocatable assembler-language expressions.

CONVDATA(*data-area***)**
specifies the 24-byte application data area into which conversation-related information is to be moved. A description of the format of the data area is given in the discussion of CONVDATA fields in the Testing indicators.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed GDS ALLOCATE command, or the token representing the principal session (returned by a previously executed GDS ASSIGN command).

RETCODE(*data-area***)**
specifies the 6-byte application data area into which return code information (shown in Table 14) is to be moved.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

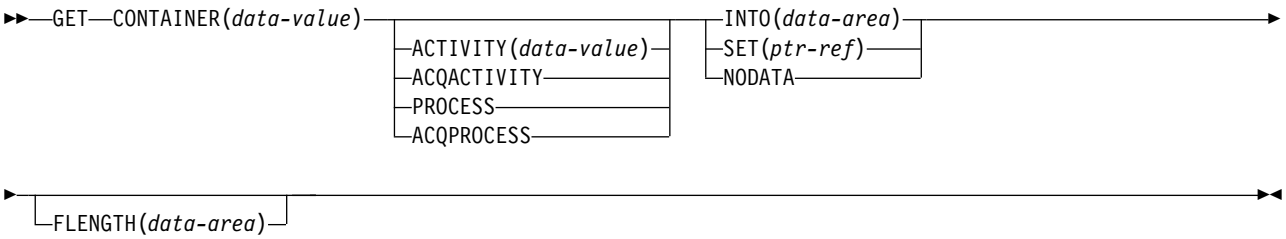
Table 14. GDS WAIT return codes

RETCODE (hexadecimal)	Description
03 00	CONVID is for a session that is not APPC.
03 00	CONVID is for a session that is in use by CPI Communications.
03 04	CONVID is for a conversation that is not basic.
03 08	A state check has occurred.
04	CONVID is for a session that is not allocated to task, or that is a relay link.

GET CONTAINER (BTS)

Retrieve data from a named BTS data-container.

GET CONTAINER



Conditions: ACTIVITYERR, CONTAINERERR, INVREQ, IOERR, LENGERR, LOCKED, PROCESSBUSY

Description

GET CONTAINER reads the data associated with a specified BTS activity or process into working storage.

The container which holds the data is identified by name and by the process or activity for which it is a container—the process or activity that "owns" it. The activity that owns the container can be identified:

- Explicitly, by specifying one of the PROCESS- or ACTIVITY-related options.
- Implicitly, by omitting the PROCESS- and ACTIVITY-related options. If these are omitted, the current activity is implied.

See also "PUT CONTAINER (BTS)" on page 354 and "MOVE CONTAINER (BTS)" on page 343.

Options

ACQACTIVITY
specifies either of the following:

- If the program that issues the command has acquired a process, that the container is owned by the root activity of that process.
- Otherwise, that the container is owned by the activity that the program has acquired by means of an ACQUIRE ACTIVITYID command.

ACQPROCESS
specifies that the container is owned by the process that the program that issues the command has acquired in the current unit of work.

ACTIVITY(data-value)
specifies the name (1–16 characters) of the activity that owns the container. This must be a child of the current activity.

CONTAINER(data-value)
specifies the name (1–16 characters) of the container that holds the data to be retrieved.

FLENGTH(data-area)
As an input field, FLENGTH specifies, as a fullword binary value, the length of the data to be read. As an output field, FLENGTH returns the length of the

data in the container. Whether FLENGTH is an input or an output field depends on which of the INTO, SET, or NODATA options you specify.

INTO option specified
FLENGTH is both an input and an output field.

On input, FLENGTH specifies the maximum length of the data that the program accepts. If the value specified is less than zero, zero is assumed. If the length of the data exceeds the value specified, the data is truncated to that value and the LENGERR condition occurs. If the length of the data is less than the value specified, the data is copied with no padding and the LENGERR condition occurs.

FLENGTH need not be specified if the length can be generated by the compiler from the INTO variable. If you specify both INTO and FLENGTH, FLENGTH specifies the maximum length of the data that the program accepts.

On **output** (that is, on completion of the retrieval operation) CICS sets the data area, if specified, to the actual length of the data in the container.

SET or NODATA option specified
FLENGTH is an output-only field. It must be specified and specified as a data-area.

On completion of the retrieval operation, the data area is set to the actual length of the data in the container.

INTO(data-area)
specifies an area of working storage into which the retrieved data is to be placed.

NODATA
specifies that no data is to be retrieved. Use this option to discover the length of the data in the container (returned in FLENGTH).

PROCESS
specifies that the container to be retrieved is owned by the current process—that is, the process that the program that issues the command is executing on behalf of.

SET(ptr-ref)
specifies a data area in which the address of the retrieved data is returned. The data area is maintained by CICS until a subsequent GET CONTAINER command with the SET option is issued by the task, or until the task ends.

If your application needs to keep the data it should move it into its own storage.

Conditions

109 ACTIVITYERR
RESP2 values:

8 The activity named on the ACTIVITY option could not be found.

110 CONTAINERERR
RESP2 values:

10 The container named on the CONTAINER option could not be found.

16 INVREQ
RESP2 values:

2 The INTOCCSID option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one.) INTOCCSID is valid only on GET CONTAINER commands that specify (explicitly or implicitly) a channel. It is not valid on GET CONTAINER (BTS) commands.

4 The command was issued outside the scope of a currently-active activity.

15 The ACQPROCESS option was used, but the unit of work that issued the request has not acquired a process.

24 The ACQACTIVITY option was used, but the unit of work that issued the request has not acquired an activity.

25 The PROCESS option was used, but the command was issued outside the scope of a currently-active process.

17 IOERR
RESP2 values:

30 An input/output error has occurred on the repository file.

31 The record on the repository file is in use.

22 LENGERR
RESP2 values:

11 The length of the program area is not the same as the length of the data in the container. If the area is smaller, the data is truncated to fit into it. If the area is larger, the data is copied to the program area but no padding is added.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

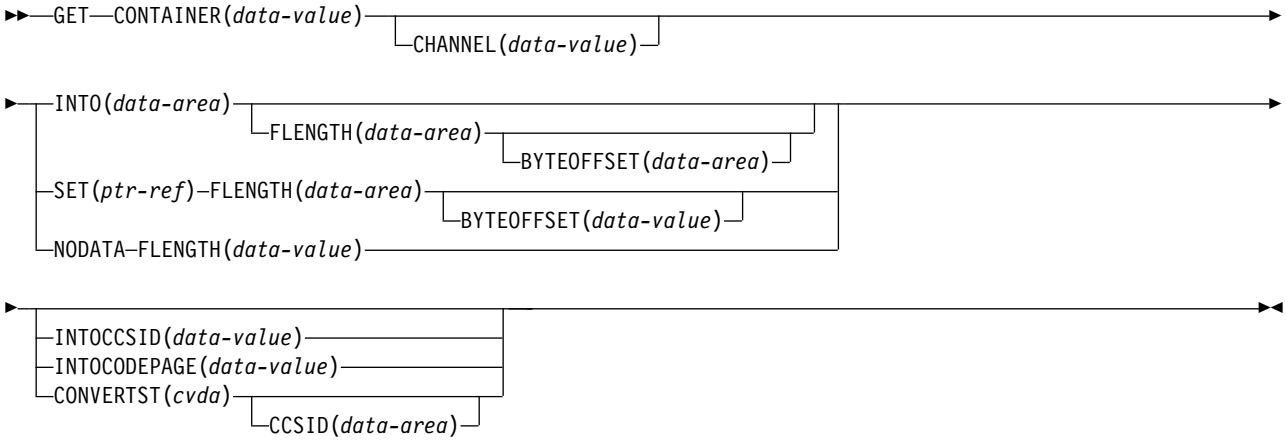
106 PROCESSBUSY
RESP2 values:

13 The request could not be satisfied because the process record is locked by another task.

GET CONTAINER (CHANNEL)

Retrieve data from a named channel container.

GET CONTAINER (CHANNEL)



Conditions: CCSIDERR, CHANNELERR, CODEPAGEERR, CONTAINERERR, INVREQ, LENGERR

This command is threadsafe.

Description

GET CONTAINER (CHANNEL) reads the data associated with a specified channel container.

The container that holds the data is identified by name and by the channel for which it is a container; the channel that "owns" it.

The channel that owns the container can be identified explicitly, by specifying the CHANNEL option, or implicitly, by omitting the CHANNEL option. When this option is omitted, the current channel is implied. If there is no current channel, the command is assumed to be an **EXEC CICS GET CONTAINER** (BTS) command. For more information, see "GET CONTAINER (BTS)" on page 232.

Options

BYTEOFFSET(*data-value*)
Specifies the offset in bytes where the data returned starts. For CHAR containers, the BYTEOFFSET value is used as an offset into the data in the requested code page. If you use a code page with multibyte characters, depending on the BYTEOFFSET value you specify, the data returned might have partial characters at the beginning, end, or both. In this situation, your application program must be able to handle and interpret the data returned. If the value specified is less than zero, zero is assumed.

CCSID(*data-area*)
Returns a fullword that contains the Coded Character Set Identifier (CCSID) of the data returned by the CONVERTST(NOCONVERT) option. You can use this option to retrieve containers with a DATATYPE of CHAR, without converting the data. If a DATATYPE of BIT is specified for the container, this value is zero.

CHANNEL(*data-value*)
Specifies the name (1 - 16 characters) of the channel that owns the container. You can specify the channel name DFHTRANSACTION to use the transaction channel.

CONTAINER(*data-value*)
Specifies the name (1 - 16 characters) of the container that holds the data to be retrieved.

CONVERTST(*cvda*)
Specifies the required data conversion status.

NOCONVERT
The container data is retrieved without being converted. If you used the WEB RECEIVE to store the HTTP body in a container, and you need to retrieve the body unconverted from that container, you must use the NOCONVERT option.

FLENGTH(*data-area*)
As an input field, FLENGTH specifies, as a fullword binary value, the length of the data to be read. As an output field, FLENGTH returns the length of the data in the container. FLENGTH is an input or an output field depending on which of the BYTEOFFSET, INTO, SET, or NODATA options you specify.

BYTEOFFSET option specified
FLENGTH is both an input and an output field.

On **input**, FLENGTH specifies the maximum length of the data that the program accepts. The data returned begins at the offset specified by the BYTEOFFSET value. If the value specified is less than zero, zero is assumed.

On **output** (that is, on completion of the retrieval operation) CICS sets the data area to the length of the data returned. The maximum length returned is equal to the length of the data in the container minus the BYTEOFFSET value.

INTO option specified
FLENGTH is both an input and an output field.

On **input**, FLENGTH specifies the maximum length of the data that the program accepts. If the value specified is less than zero, zero is assumed. If the length of the data exceeds the value specified, the data is truncated to that value and the LENGERR condition occurs. If the length of the data is less than the specified value, the data is copied but no padding is performed.

You do not need to specify FLENGTH if the length can be generated by the compiler from the INTO variable. If you specify both INTO and FLENGTH, FLENGTH specifies the maximum length of the data that the program accepts.

On **output** (that is, on completion of the retrieval operation) CICS sets the data area, if specified, to the actual length of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the data after conversion.

SET or NODATA option specified
FLENGTH is an output-only field. It must be present and must be specified as a data-area.

On completion of the retrieval operation, the data area is set to the actual length of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the data after conversion.

INTO(*data-area*)
Specifies the data area into which the retrieved data is placed.

INTOCCSID(*data-value*)
Specifies the Coded Character Set Identifier (CCSID) into which the character data in the container is converted, as a fullword binary number. If you prefer to specify an IANA name for the code page, or if you prefer to specify the CCSID as alphanumeric characters, use the INTOCODEPAGE option instead.

For CICS Transaction Server for z/OS applications, the CCSID is typically an EBCDIC CCSID. However, it is possible to specify an ASCII CCSID if, for example, you want to retrieve ASCII data without it being automatically converted to EBCDIC.

If INTOCCSID and INTOCODEPAGE are not specified, the value for conversion defaults to the CCSID of the region. The default CCSID of the region is specified on the **LOCALCCSID** system initialization parameter.

Only character data can be converted, and only then if a DATATYPE of CHAR was specified on the **PUT CONTAINER** or **PUT64 CONTAINER** command used to place the data in the container. A DATATYPE of CHAR is implied if FROMCCSID or FROMCODEPAGE is specified on the **PUT CONTAINER** or **PUT64 CONTAINER** command.

For more information about data conversion with channels, see Data conversion with channels.

For an explanation of CCSIDs, see Preparing for code page conversion with channels.

INTOCODEPAGE(*data-value*)
Specifies an IANA-registered alphanumeric charset name or a Coded Character Set Identifier (CCSID) for the code page into which the character data in the container is to be converted, using up to 40 alphanumeric characters, including appropriate punctuation. Use this option instead of the CCSID option if you prefer to use an IANA-registered charset name, as specified in the Content-Type header for an HTTP request. CICS converts the IANA name into a CCSID, and the subsequent data conversion process is identical. Also use this option if you prefer to specify the CCSID in alphanumeric characters, rather than as a fullword binary number.

Where an IANA name exists for a code page and CICS supports its use, the name is listed with the CCSID. For more information, see Preparing for code page conversion with channels.

NODATA
Specifies that no data is retrieved. Use this option to discover the length of the data in the container (returned in FLENGTH).

The length of character data may change if data conversion takes place. Therefore, if character data is to be converted into any CCSID *other than that of this region*, when you specify NODATA you should also specify INTOCCSID. This ensures that the correct length of the converted data is returned in FLENGTH.

SET(*ptr-ref*)
Specifies a data area in which the address of the retrieved data is returned.

If the application program that issues the **GET CONTAINER** command is defined with DATALOCATION(ANY), the address of the data can be above or below the 16 MB line. If the application program is defined with DATALOCATION(BELOW), the address of the data is below the 16 MB line. If

TASKDATAKEY(USER) is specified for the executing transaction, the data returned is in user key; otherwise it is in CICS key.

CICS maintains the data area until any of the following occurs:

- A subsequent **GET CONTAINER** or **GET64 CONTAINER** command with the SET option, for the same container in the same channel, is issued by any program that can access this storage.
- The container is deleted by a **DELETE CONTAINER** command.
- The container is moved by a **MOVE CONTAINER** command.
- The channel goes out of program scope.
- The channel, and the containers that are in it, are deleted by a **DELETE CHANNEL** command.

Beware of linking to other programs that might issue one of these commands.

Do not issue a **FREEMAIN** command to release this storage.

If your application needs to keep the data, it should move it into its own storage.

Conditions

123 CCSIDERR
RESP2 values:

1 The CCSID specified on the INTOCCSID option is outside the range of valid CCSID values.

2 The CCSID specified on the INTOCCSID option and the CCSID of the container are an unsupported combination. (The CCSID of the container is the value that was specified using either FROMCODEPAGE or FROMCCSID, or defaulted, when the container was built.)

3 The data was created with a data type of BIT. Code page conversion is not possible. The data was returned without any code page conversion.

4 One or more characters could not be converted. The character has been replaced by a blank in the converted data.

5 There was an internal error in the code page conversion of a container.

122 CHANNELERR
RESP2 values:

2 The channel specified on the CHANNEL option could not be found.

125 CODEPAGEERR
RESP2 values:

1 The code page specified on the INTOCODEPAGE option is not supported.

2 The code page specified on the INTOCODEPAGE option and the code page of the channel are an unsupported combination.

3 The data was created with a data type of BIT. Code page conversion is not possible. The data was returned without any code page conversion.

4 One or more characters could not be converted. The character has been replaced by a blank in the converted data.

5 There was an internal error in the code page conversion of a container.

110 CONTAINERERR
RESP2 values:

10 The container named on the CONTAINER option could not be found.

16 INVREQ
RESP2 values:

2 The INTOCCSID option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one.) INTOCCSID is valid only on GET CONTAINER commands that specify (explicitly or implicitly) a channel.

4 The CHANNEL option was not specified, there is no current channel (because the program that issued the command was not passed one), and the command was issued outside the scope of a currently-active BTS activity.

5 The CONVERTST cvda value is invalid.

22 LENGERR
RESP2 values:

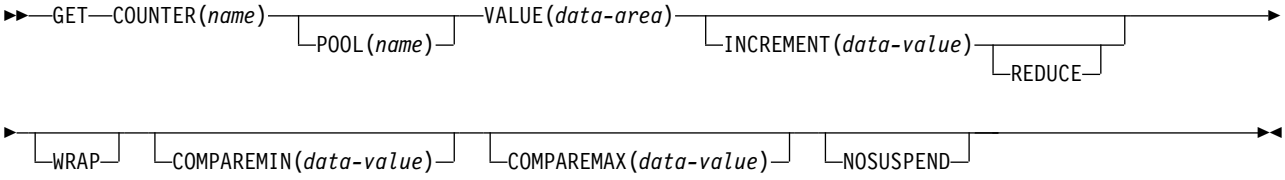
11 The length of the program area is shorter than the length of the data in the container. When the area is smaller, the data is truncated to fit into it.

12 The offset is greater than, or equal to, the length of the container.

GET COUNTER and GET DCOUNTER

Get the next number from the named counter in the specified pool. Use COUNTER for fullword signed counters and DCOUNTER for doubleword unsigned counters.

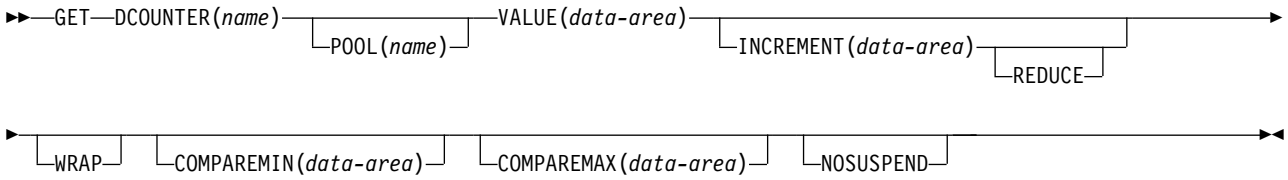
GET COUNTER



Conditions: BUSY, INVREQ, LENGERR, SUPPRESSED

This command is threadsafe.

GET DCOUNTER



Conditions: BUSY, INVREQ, LENGERR, SUPPRESSED

This command is threadsafe.

Description

These counter commands obtain, from the named counter server, the current number from the named counter in the specified pool, and updates the current number by the default, or by a specified, increment. The default increment is 1.

You can use the COMPAREMAX and COMPAREMIN options to obtain a number only if it falls within a specified range, or is above or below a specified value.

For information about specifying fullword and doubleword variables on these named counter commands, see "CICS command argument values" on page 3.

Options

COMPAREMAX(*data-value***)**
Specifies, as a fullword signed binary value (or doubleword unsigned binary value for DCOUNTER), a value to compare with the current value of the named counter, and makes the result of the GET command conditional on the comparison:

- If the current value to assign is less than, or equal to, the value specified on the COMPAREMAX parameter, the current value is returned, and the response is normal.
- If the current value is greater than the specified value, CICS returns an exception condition.

Normally, the COMPAREMAX value is greater than the COMPAREMIN value and the current value must satisfy both comparisons (that is, it must be between the two values or equal to one of them).

You can specify a COMPAREMAX value that is less than the COMPAREMIN value. In this situation, the current value is considered to be in range if it satisfies either the COMPAREMIN or the COMPAREMAX comparison.

COMPAREMIN(*data-value***)**
Specifies, as a fullword signed binary value (or doubleword unsigned binary value for DCOUNTER), a value to compare with the current value of the named counter, and makes the result of the GET command conditional on the comparison:

- If the current value to assign is equal to, or greater than, the value specified on the COMPAREMIN parameter, the current value is returned, and the response is normal.
- If the current value is less than the specified value, CICS returns an exception condition.

Note: You can specify a COMPAREMIN value that is greater than the COMPAREMAX value. See the COMPAREMAX parameter for the effect of this.

COUNTER(*name***)**
Specifies the name of the fullword counter from which the current number is to be assigned to the application program. The name can be up to 16 alphanumeric characters. If *name* is a variable that contains a name that is less than 16 characters, the name must be padded with trailing blanks.

DCOUNTER(*name***)**
Specifies the name of the doubleword counter from which the current number is to be assigned to the application program. The name can be up to 16 alphanumeric characters. If *name* is a variable that contains a name that is less than 16 characters, the name must be padded with trailing blanks.

INCREMENT(*data-value***)**
Specifies, as a fullword signed binary value (or doubleword unsigned binary value for DCOUNTER), an increment by which the named counter is to be updated, instead of the default value of 1. The counter is incremented after the current number has been assigned.

Specifying an increment to override the default increment of 1 enables the application program to obtain exclusive use of more than one number for each call. For example, to obtain exclusive use of a block of 20 numbers, specify INCREMENT(20).

See the description of the REDUCE and WRAP options for the effect of specifying an increment when the counter is at, or near, the maximum value.

NOSUSPEND
Specifies an immediate return to the application program during a coupling facility structure rebuild. If you omit the NOSUSPEND parameter, the request is retried until it is successful. If you specify NOSUSPEND, the request fails immediately, and CICS returns BUSY, RESP2=500.

POOL(*poolname***)**
Specifies an 8-character string to use as a pool selection parameter to select the pool in which the named counter resides. The string can be a logical pool name, or the actual pool name.

Valid characters for the pool selector string are A through Z, 0 through 9, $ @ # and _ (underscore). If *name* is a variable that contains a name that is less than 8 characters, the name must be padded with trailing blanks.

This parameter is optional. If you omit the name of the pool, a pool selector value of 8 blanks is assumed.

If there is no matching entry in the DFHNCOPT options table, CICS uses the name specified on the NCPLDFT system initialization parameter, which specifies the default named counter pool.

For information about generating a named counter options table using the DFHNCO macro, see .

REDUCE
Specifies that you want the named counter server to reduce the specified increment if the range of numbers remaining to be assigned is too small.

The range of numbers is too small if the difference between the current value and the maximum value plus 1 is less then the specified increment, in which case:

- If you specify REDUCE, the INCREMENT parameter value is reduced and the GET request succeeds. In this case, the GET command has reserved a range of numbers less than that specified by the INCREMENT parameter, and the current value is updated to the maximum value plus 1.
- If you do not specify the REDUCE option, the result depends on whether you specify the WRAP option. If the REDUCE and WRAP options are both omitted, the request fails with the counter-at-limit error (SUPPRESSED, RESP2=101), but the current number is not changed. For example, if a request specifies an INCREMENT parameter value of 15 when the current number is 199 990 and the counter maximum number is defined as 199 999, the GET command fails because updating the counter by the specified increment would cause the current number to exceed 200 000.

VALUE(*data-area***)**
Specifies the data area (fullword signed data-area for COUNTER, and doubleword unsigned data-area for DCOUNTER) into which CICS returns the current number, obtained from the named counter server for the specified pool.

WRAP
Specifies that you want the named counter server to rewind the named counter automatically if it is in a counter-at-limit condition, thus avoiding the error condition that would otherwise result.

If the named counter is in the counter-at-limit condition, or the increment specified without the REDUCE option would cause the counter-at-limit condition, the counter server acts as follows:

- It resets the current value of the named counter equal to the minimum value defined for the counter.
- It returns the new current value to the application program, with DFHRESP(NORMAL).
- It updates the current value by the required increment ready for the next request.

If you omit the WRAP option, and the counter-at-limit condition is reached, CICS returns SUPPRESSED, RESP2=101.

Conditions

128 BUSY
RESP2 values:

500 The NOSUSPEND option was specified on the command, and the coupling facility structure is not currently available during a rebuild.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

201 Named counter not found.

301 The server has reported an error code that is not understood by the named counter interface. Generally, this is not possible unless the interface load module, DFHNCIF, is at a lower maintenance or release level than the server itself.

303 An unexpected error, such as structure failure or loss of connectivity, has occurred on a macro used to access the coupling facility. Further information is in message DFHNC0441 in the application job log.

304 The pool selection parameter specified in the program cannot be resolved to a valid server name using the current options table.

305 The interface cannot establish a connection to the server for the selected named counter pool. Further information is in an AXM services message (AXMSC*nnnn*) in the application job log.

306 An abend occurred during server processing of a request. Further information is in a message in the application job log and the server job log.

308 The DFHNCOPT options table module, required to resolve a pool name, cannot be loaded.

309 During processing of the options table, the named counter interface encountered an unknown entry format. Either the options table is not correctly generated, or the DFHNCIF interface load module is not at the same release level as the options table.

310 An options table entry that matches the given pool name specified a user exit program, but the user exit program is not link-edited with the options table and cannot be loaded.

311 A response from the named counter server to the client region interface module, DFHNCIF, indicates that a system-managed rebuild is in progress but the EXEC CICS interface does not recognize the condition. This means that the CICS region is at CICS TS 2.1 or earlier.

403 The POOL parameter contains invalid characters or embedded spaces.

404 The COUNTER parameter contains invalid characters or embedded spaces.

406 The INCREMENT value is invalid. The value specified cannot be greater than the total range of the counter ((maximum value - minimum value) + 1).

Default action: terminate the task abnormally.

22 LENGERR

LENGERR occurs for COUNTER commands only and does not apply to DCOUNTER requests. It occurs when a counter that was defined by a DCOUNTER command or by the CALL interface has a value that is too large to be correctly represented as a fullword signed binary value (that is, the counter uses more than 31 bits).

In each of the three cases of overflow, the named counter server completes the operation, and returns a warning response to CICS, which CICS returns to your application program as the RESP2 value. The data area contains the low-order 32 bits returned from the named counter server, which could be a negative number.

RESP2 values:

001 The current value that the server has attempted to return in the VALUE data area has overflowed into the high-order (sign) bit (that is, the value returned is negative).

002 The current value is too large for a fullword data area by only 1 bit. In this case, the overflow value is exactly 1.

003 The current value is too large for a fullword data area by a value greater than 1.

Default action: terminate the task abnormally.

72 SUPPRESSED
RESP2 values:

101 The maximum value for the named counter has already been assigned and the counter is in the 'counter-at-limit' condition. No more counter numbers can be assigned until the named counter has been reset, either by a REWIND command, or by specifying the WRAP option on the GET command.

103 One of the following:

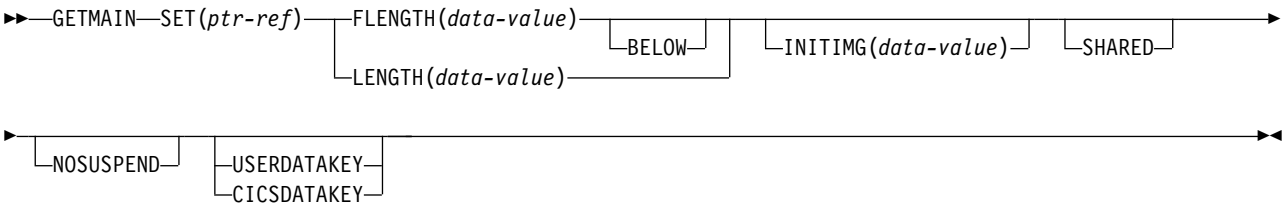
- The current value of the named counter is not within the range specified by the **COMPAREMAX** and **COMPAREMIN** parameters, when both are specified
- The current value of the named counter is greater than the **COMPAREMAX** parameter or less than the **COMPAREMIN** parameter, when only one option is specified.

Default action: terminate the task abnormally.

GETMAIN

Get main 24-bit or 31-bit storage.

GETMAIN



Conditions: LENGERR, NOSTG

This command is threadsafe.

Description

GETMAIN gets a main storage area of the size indicated by the FLENGTH option. The address of the area is returned in the pointer reference supplied in the SET option.

This command includes the LENGTH option for compatibility purposes only; for programs that run with current versions of CICS, use FLENGTH.

CICS always allocates on 16-byte boundaries and rounds the requested length up to the nearest 16-byte multiple. There is no default initialization, so if you require the storage to be initialized to a specific bit configuration, you must use the INITIMG option.

CICS allocates storage from one of the following dynamic storage areas (DSAs):

- DSAs in 24-bit storage:
  - The CICS dynamic storage area (CDSA), below 16 MB (below the line)

  - The user dynamic storage area (UDSA), below 16 MB
  - The shared dynamic storage area (SDSA), below 16 MB
- DSAs in 31-bit storage; the extended dynamic storage area (EDSA):
  - The extended CICS dynamic storage area (ECDSA), above 16 MB but below 2 GB (above the line)
  - The extended user dynamic storage area (EUDSA), above 16 MB but below 2 GB
  - The extended shared dynamic storage area (ESDSA), above 16 MB but below 2 GB

For more information about these DSAs, see CICS dynamic storage areas.

Note: You cannot use GETMAIN to get storage from the following DSAs:

- The read-only DSA (RDSA).
- The extended read-only DSA (ERDSA).
- The extended trusted DSA (ETDSA).
- DSAs in the above-the-bar dynamic storage area (GDSA). To obtain storage from these DSAs, use the GETMAIN64 command.

CICS allocates storage from 24-bit or 31-bit storage, depending on the following options:

- The FLENGTH option with BELOW also specified. CICS obtains the storage from a DSA in 24-bit storage.
- The FLENGTH option, BELOW not specified. The AMODE of the requesting program determines the location of the obtained storage. For example, for an AMODE(31) program, CICS obtains the storage from the EDSA.
- The LENGTH option. CICS obtains the storage from a DSA in 24-bit storage.

CICS allocates storage from a CICS-key, user-key, or shared DSA, depending on the following options:

- USERDATAKEY
- CICSDATAKEY
- SHARED
- If no data-key option is specified on the GETMAIN command, the TASKDATAKEY option on the RDO TRANSACTION resource definition under which the requesting program is running. See TRANSACTION attributes.

The effect of the data-key options is summarized in the following table:

Table 15. Data-key options on GETMAIN command

No data-key option	USERDATAKEY specified	CICSDATAKEY specified
Storage key is determined by TASKDATAKEY on transaction definition	User-key storage. Storage is obtained from the UDSA or EUDSA if the SHARED option is not specified, or from the SDSA or ESDSA if the SHARED option is specified.	CICS-key storage. Storage is obtained from the CDSA or ECDSA.

The data-key option on the GETMAIN command overrides the TASKDATAKEY option on the RDO TRANSACTION resource definition. For example, you can

specify CICSDATAKEY to ensure that the requesting program obtains CICS-key storage from a CICS DSA, even if TASKDATAKEY(USER) is specified on the RDO TRANSACTION resource definition.

The storage that a task gets is available until it is released with a FREEMAIN or FREEMAIN64 command. For an area that is obtained without the SHARED option, only the task that acquired the storage can release it, and at task end CICS automatically releases such storage not already released.

Any storage acquired with the SHARED option is accessible by all tasks, including those that are running with transaction isolation. However, a SHARED area is not released at task end and remains until explicitly freed; any task can issue the FREEMAIN or FREEMAIN64 request. This means that you can use SHARED storage in task-to-task communication.

Note: Using the GETMAIN command with the SHARED option could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. For more information about transaction affinities, see Affinity .

Options

BELOW
Specifies that 24-bit (below 16 MB) storage is obtained; that is, from the CDSA, UDSA, or SDSA.

CICSDATAKEY
Specifies that CICS allocates storage from a CICS-key DSA (CDSA or ECDSA), overriding the TASKDATAKEY option specified on the transaction resource definition. If you do not specify a data-key option, the storage key (CICS-key or user-key) depends on the TASKDATAKEY option on the transaction resource definition.

Note: If the program is running under a task defined with TASKDATAKEY(USER) on the transaction resource definition, do not explicitly use FREEMAIN or FREEMAIN64, but allow the storage to be freed as part of the task termination.

FLENGTH(*data-value***)**
Specifies the number of bytes of storage required, in fullword binary format.

The maximum length that you can specify is the value of the limit for the corresponding DSA; that is, DSALIMIT or EDSALIMIT. These are the system initialization parameters that define the overall storage limit within which CICS can allocate and manage the individual DSAs in 24-bit and 31-bit storage, respectively.

If the length requested is greater than the corresponding DSALIMIT or EDSALIMIT value, the LENGERR condition occurs. If the length requested is less than the corresponding limit, but is greater than the available storage, a NOSTG condition occurs.

INITIMG(*data-value***)**
Specifies an optional 1-byte initialization value. If you specify INITIMG, CICS sets every byte of the acquired storage to the bit string you provide. Otherwise, CICS does not initialize the storage. In COBOL programs only, you must use a data area rather than a data value to define the initialization bit string.

LENGTH(*data-value***)**
This option is supported only for compatibility with programs that are written to run under earlier releases of CICS. It is advisable to use FLENGTH.

LENGTH specifies the number of bytes (unsigned halfword binary value) of storage required. This option implies storage from below the 16 MB line and has an upper limit of 65520 bytes. For a larger area or storage above 16 MB, use FLENGTH.
If the value of LENGTH is zero, a LENGERR condition occurs. If the length requested is greater than the available storage, a NOSTG condition occurs.

NOSUSPEND
Specifies that if no storage is available, CICS does not suspend the task, but issues the NOSTG condition.

If a HANDLE CONDITION for NOSTG is active when the command is executed, control is passed to the user label supplied in the HANDLE CONDITION. This takes precedence over the NOSUSPEND option, but is deactivated by NOHANDLE or RESP.

SET(*ptr-ref***)**
Sets the pointer reference to the address of the acquired main storage. The pointer is set to the first byte of the storage area.

SHARED
Prevents the automatic release of storage obtained by a GETMAIN command at the end of the task that requested it. This enables task-to-task communication. An area obtained with SHARED is not released until a corresponding FREEMAIN is issued, whether by the requesting task or some other task.

Be aware that if a task abends, any shared storage acquired is not automatically released.

USERDATAKEY
Specifies that CICS allocates storage from a user-key DSA (the UDSA, SDSA, EUDSA, or ESDSA), overriding the TASKDATAKEY option specified on the transaction resource definition. If you do not specify a data-key option, the storage key (CICS-key or user-key) depends on the TASKDATAKEY option on the transaction resource definition.

Conditions

22 LENGERR
RESP2 values:

1 The FLENGTH value is less than 1 or greater than the length of the target dynamic storage area from which the request is to be satisfied. See the discussion about DSAs in CICS dynamic storage areas.

Also occurs if the LENGTH value is zero.

Default action: terminate the task abnormally.

42 NOSTG
RESP2 values:

2 The storage requested is more than is currently available in the target DSA. See the discussion about DSAs in CICS dynamic storage areas.

Default action: ignore the condition. An active HANDLE CONDITION NOSTG also raises this condition.

Examples

The following example shows how to get a 1024-byte area from user-key storage below 16 MB (assuming that TASKDATAKEY(USER) is specified on the RDO TRANSACTION resource definition), and initialize it to spaces:

```
EXEC CICS GETMAIN SET(PTR)
          FLENGTH(1024)
          BELOW
          INITIMG(BLANK)
```

You must define BLANK in your program as the character representing a space.

The following example shows how to get a 2048-byte area from CICS-key storage above 16 MB but below 2 GB (regardless of the TASKDATAKEY option specified on the transaction resource definition), and initialize it to spaces:

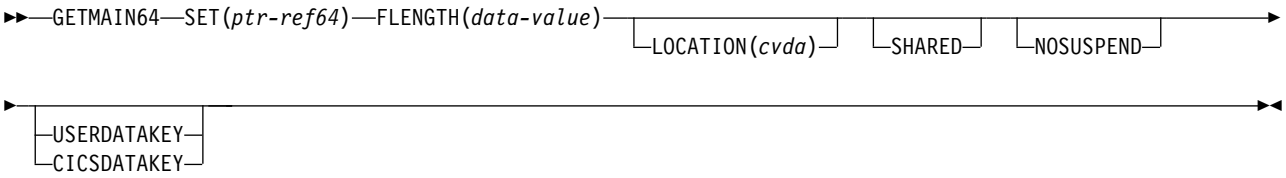
```
EXEC CICS GETMAIN SET(PTR)
          FLENGTH(2048)
          INITIMG(BLANK)
          CICSDATAKEY
```

GETMAIN64

Get 24-bit, 31-bit, or 64-bit storage. This command is for use only in non-Language Environment (LE) AMODE(64) assembler language application programs.

See Assembler language programming restrictions and requirements.

GETMAIN64



Conditions: INVREQ, LENGERR, NOSTG

This command is threadsafe.

Description

GETMAIN64 gets a main storage area of the size indicated by the FLENGTH option. The address of the area is returned in the 64-bit pointer reference supplied in the SET option. By default, the addressing mode (AMODE) of the requesting program determines the location of the obtained storage. Therefore, for an AMODE(64) program, GETMAIN64 obtains 64-bit storage.

You can also use the LOCATION parameter to specify that 24-bit or 31-bit storage is obtained, regardless of the AMODE of the calling program. For example, if you specify LOCATION(LOC31), a 64-bit address for an area of 31-bit storage is returned.

CICS always allocates on 16-byte boundaries and rounds the requested length up to the nearest 16-byte multiple.

CICS allocates storage from one of the following dynamic storage areas (DSAs):

- DSAs in 24-bit storage:
  - The CICS dynamic storage area (CDSA), below 16 MB (below the line)
  - The user dynamic storage area (UDSA), below 16 MB
  - The shared dynamic storage area (SDSA), below 16 MB
- DSAs in 31-bit storage; the extended dynamic storage area (EDSA):
  - The extended CICS dynamic storage area (ECDSA), above 16 MB but below 2 GB (above the line)
  - The extended user dynamic storage area (EUDSA), above 16 MB but below 2 GB
  - The extended shared dynamic storage area (ESDSA), above 16 MB but below 2 GB
- DSAs in 64-bit storage: the above-the-bar dynamic storage area (GDSA):
  - The above-the-bar CICS DSA (GCDSA)
  - The above-the-bar user DSA (GUDSA)
  - The above-the-bar shared DSA (GSDSA)

For more information about these DSAs, see CICS dynamic storage areas.

Note: You cannot use GETMAIN64 to obtain storage from the following DSAs:

- The read-only DSA (RDSA)
- The extended read-only DSA (ERDSA)
- The extended trusted DSA (ETDSA)

CICS allocates storage from a CICS-key, user-key, or shared DSA, depending on the following options:

- USERDATAKEY
- CICSDATAKEY
- SHARED
- If no data-key option is specified on the GETMAIN64 command, the TASKDATAKEY option on the RDO TRANSACTION resource definition under which the requesting program is running. See TRANSACTION attributes.

The following table summarizes the effect of the data-key options.

Table 16. Data-key options on GETMAIN64 command

No data-key option	USERDATAKEY specified	CICSDATAKEY specified
Storage key is determined by TASKDATAKEY on the transaction definition	User-key storage. If the SHARED option is not specified, storage is obtained from the UDSA, EUDSA, or GUDSA. If the SHARED option is specified, storage is obtained from the SDSA, ESDSA, or GSDSA.	CICS-key storage. Storage is obtained from the CDSA, ECDSA, or GCDSA.

The data-key option on the GETMAIN64 command overrides the TASKDATAKEY option on the RDO TRANSACTION resource definition. For example, you can

specify CICSDATAKEY to ensure that the requesting program obtains CICS-key storage from a CICS DSA, even if TASKDATAKEY(USER) is specified on the RDO TRANSACTION resource definition.

The storage that a task obtains is available until it is released with a FREEMAIN or FREEMAIN64 command. For storage that is obtained without the SHARED option, only the task that acquired the storage can release it. At task end, CICS automatically releases such storage not already released.

Any storage obtained with the SHARED option is accessible by all tasks, including those that are running with transaction isolation. However, a shared area is not released at task end and remains until explicitly freed; any task can issue the FREEMAIN or FREEMAIN64 request. This means that you can use shared storage in task-to-task communication.

Note: Using the GETMAIN64 command with the SHARED option could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. For more information about transaction affinities, see Affinity .

Options

CICSDATAKEY
Specifies that CICS allocates storage from a CICS-key DSA (CDSA, ECDSA, or GCDSA), overriding the TASKDATAKEY option specified on the transaction resource definition. If you do not specify a data-key option, the storage key (CICS-key or user-key) depends on the TASKDATAKEY option on the transaction resource definition.

Note: If the program is running under a task defined with TASKDATAKEY(USER) on the transaction resource definition, do not explicitly use a FREEMAIN or FREEMAIN64 request, but allow the storage to be freed as part of the task termination.

FLENGTH(*data-value***)**
Specifies the number of bytes of storage required, in fullword binary format.

For 64-bit storage, the maximum length that you can specify is 2146435056 (2 GB - (1 MB + 16 bytes)). If the length requested is greater than the available storage, a NOSTG condition occurs.

For 24-bit or 31-bit storage, the maximum length that you can specify is the value of the limit for the corresponding DSA, that is, DSALIMIT or EDSALIMIT. DSALIMIT and EDSALIMIT are system initialization parameters that define the overall storage limits within which CICS can allocate and manage the individual DSAs in 24-bit and 31-bit storage, respectively. If the length requested is bigger than the corresponding DSALIMIT or EDSALIMIT value, the LENGERR condition occurs. If the length requested is less than the corresponding limit, but is greater than the available storage, a NOSTG condition occurs.

LOCATION(*cvda***)**
Specifies that CICS allocates storage from 24-bit or 31-bit storage, regardless of the AMODE of the calling program. CVDA values are as follows:

LOC24
24-bit storage (below 16 MB) is obtained; that is, from the CDSA, UDSA, or SDSA.

LOC31
31-bit storage (above 16 MB but below 2 GB) is obtained; that is, from the ECDSA, EUDSA, or ESDSA.

NOSUSPEND
Specifies that if no storage is available, CICS does not suspend the task, but issues the NOSTG condition.

SET(*ptr-ref64***)**
Sets the 64-bit pointer reference to the address of the acquired main storage. The pointer is set to the first byte of the storage area.

The pointer reference returns a 64-bit address for an area of 64-bit, 31-bit, or 24-bit storage. The location of the obtained storage depends on the AMODE of the caller, unless a LOCATION option is specified.

SHARED
Prevents the automatic release of storage obtained by a GETMAIN64 command at the end of the task that requested it. This enables task-to-task communication. An area obtained with SHARED is not released until a corresponding FREEMAN or FREEMAIN64 request is issued, whether by the requesting task or another task.

Be aware that if a task abends, any shared storage acquired is not automatically released.

USERDATAKEY
Specifies that CICS allocates storage from a user-key DSA (UDSA, SDSA, EUDSA, ESDSA, GUDSA, or GSDSA), overriding the TASKDATAKEY option specified on the transaction resource definition. If you do not specify a data-key option, the storage key (CICS-key or user-key) depends on the TASKDATAKEY option on the transaction resource definition.

Conditions

16 INVREQ
RESP2 values:

3 The LOCATION option is not valid.

22 LENGERR
RESP2 values:

1 The FLENGTH value is less than 1, or greater than the length of the target dynamic storage area from which the request is to be satisfied. See the discussion about DSAs in CICS dynamic storage areas.

42 NOSTG
RESP2 values:

2 The storage requested is more than is currently available in the target DSA. See the discussion about DSAs in CICS dynamic storage areas.

Example

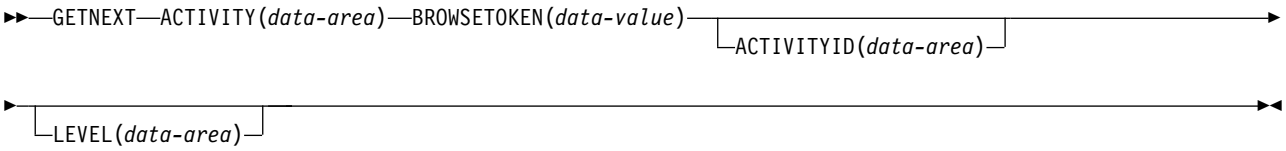
The following example shows how to get a 1048576 byte area from CICS-key storage above the bar (regardless of the TASKDATAKEY option specified on the transaction resource definition):

```
EXEC CICS GETMAIN64 SET(5)
          FLENGTH(1048576)
          CICSDATAKEY
```

GETNEXT ACTIVITY

Browse the child activities of a BTS activity, or the descendant activities of a BTS process.

GETNEXT ACTIVITY



Conditions: ACTIVITYERR, END, ILLOGIC, IOERR, TOKENERR

Description

GETNEXT ACTIVITY returns either:

- The name and identifier of the next child activity of a BTS activity (if the PROCESS and PROCESSTYPE options were omitted from the STARTBROWSE ACTIVITY command)
- The name and identifier of the next descendant activity of a BTS process (if the PROCESS and PROCESSTYPE options were specified on the STARTBROWSE ACTIVITY command).

You can use the INQUIRE ACTIVITYID command to query the identified activity.

Options

ACTIVITYID(data-area)
returns the 52-character identifier of the next activity.

ACTIVITY(data-area)
returns the 16-character name of the next activity.

BROWSETOKEN(data-value)
specifies, as a fullword binary value, a browse token returned on a previous STARTBROWSE ACTIVITY command.

LEVEL(data-area)
returns a fullword value indicating the depth in the activity-tree at which the next activity lies.

On a browse of the descendant activities of a process, a value of '0' indicates the root activity, '1' a child of the root activity, '2' a grandchild of the root activity, and so on.

On a browse of the child activities of an activity, the value returned is always 0.

Conditions

109 ACTIVITYERR
RESP2 values:

19 The request timed out. It may be that another task using this activity-record has been prevented from ending.

83 END
RESP2 values:

2 There are no more resource definitions of this type.

21 ILLOGIC
RESP2 values:

1 The value specified in the BROWSETOKEN option matches a current browse token, but not one that is being used for an activity browse.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

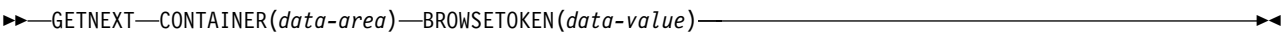
112 TOKENERR
RESP2 values:

3 The browse token is not valid.

GETNEXT CONTAINER (BTS)

Browse the containers associated with a BTS activity or process.

GETNEXT CONTAINER



Conditions: END, ILLOGIC, TOKENERR

Description

GETNEXT CONTAINER returns the name of the next container associated with a BTS activity or process. You can use the INQUIRE CONTAINER command to query the returned container.

Note:

1. You can use successive GETNEXT CONTAINER commands to retrieve the names of all the process's or activity's containers that existed at the time the STARTBROWSE CONTAINER command was executed. However, the names of any containers that are deleted after the STARTBROWSE and before they have been returned by a GETNEXT are not returned.
2. The names of any containers that are created on (or moved to) this process or activity after the STARTBROWSE command is executed may or may not be returned.
3. The order in which containers are returned is undefined.

Options

BROWSETOKEN(data-value)
specifies, as a fullword binary value, a browse token returned on a previous STARTBROWSE CONTAINER command.

CONTAINER(data-area)
returns the 16-character name of the next data-container.

Conditions

83 END
RESP2 values:

2 The are no more containers for this process or activity.

21 ILLOGIC
RESP2 values:

1 The value specified in the BROWSETOKEN option matches a current browse token, but not one that is being used for a browse of containers.

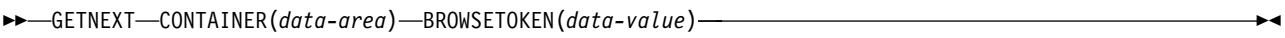
112 TOKENERR
RESP2 values:

3 The browse token is not valid.

GETNEXT CONTAINER (CHANNEL)

Browse the containers associated with a channel.

GETNEXT CONTAINER



Conditions: END, ILLOGIC, TOKENERR

Description

GETNEXT CONTAINER (CHANNEL) returns the name of the next container associated with a channel. You can use the INQUIRE CONTAINER command to query the returned container.

Note:

1. You can use successive GETNEXT CONTAINER (CHANNEL) commands to retrieve the names of all the channel's containers that existed at the time the STARTBROWSE CONTAINER (CHANNEL) command was executed. However, the names of any containers that are deleted after the STARTBROWSE and before they have been returned by a GETNEXT are not returned.
2. The names of any containers that are created on (or moved to) this channel or activity after the STARTBROWSE command is executed may or may not be returned.
3. The order in which containers are returned is undefined.

Options

BROWSETOKEN(data-value)
specifies, as a fullword binary value, a browse token returned on a previous STARTBROWSE CONTAINER (CHANNEL) command.

CONTAINER(data-area)
returns the 16-character name of the next data-container.

Conditions

83 END
RESP2 values:

2 There are no more containers.

21 ILLOGIC
RESP2 values:

1 The value specified in the BROWSETOKEN option matches a current browse token, but not one that is being used for a browse of containers.

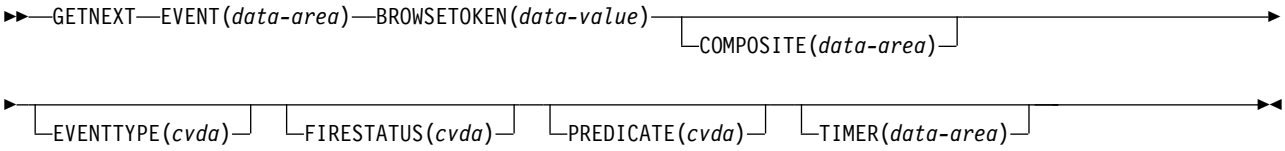
112 TOKENERR
RESP2 values:

3 The browse token is not valid.

GETNEXT EVENT

Browse the events known to a BTS activity.

GETNEXT EVENT



Conditions: END, TOKENERR

Description

GETNEXT EVENT returns the attributes of the next event, or sub-event, that is within the scope of a BTS activity.

Options

BROWSETOKEN(data-value)
specifies, as a fullword binary value, a browse token returned on a previous STARTBROWSE EVENT command.

COMPOSITE(data-area)
returns, if the named event is a sub-event, the 16-character name of the composite event that it is part of.

EVENT(data-area)
returns the 16-character name of the next event. This may be:

- An atomic event. An atomic event returned on this command may or may not be a sub-event.
- A composite event.
- A system event.

EVENTTYPE(cvda)
indicates the type of the named event. CVDA values are:

ACTIVITY
Activity completion

COMPOSITE
Composite

INPUT
Input

SYSTEM
System

TIMER
Timer.

FIRESTATUS(cvda)
indicates the state of the named event. CVDA values are:

FIRED
The event has fired normally.

NOTFIRED
The event has not fired.

PREDICATE(cvda)
indicates, if the named event is composite, the Boolean operator applied to its predicate. CVDA values are:

AND The Boolean operator applied to the predicate is AND.

OR The Boolean operator applied to the predicate is OR.

TIMER(data-area)
returns, if the named event is a timer event, the 16-character name of its associated timer.

Conditions

83 END
RESP2 values:

2 There are no more resource definitions of this type.

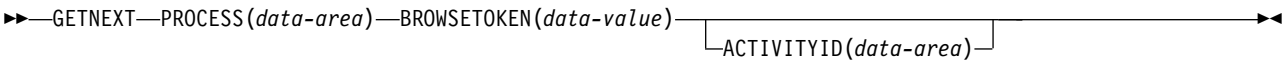
112 TOKENERR
RESP2 values:

3 The browse token is not valid.

GETNEXT PROCESS

Browse all processes of a specified type within the CICS business transaction services system.

GETNEXT PROCESS



Conditions: END, ILLOGIC, IOERR, PROCESSERR, TOKENERR

Description

GETNEXT PROCESS returns the name of the next process of a specified type within the CICS business transaction services system.

Options

ACTIVITYID(data-area)
returns the 52-character identifier of the next process's root activity.

BROWSETOKEN(data-value)
specifies, as a fullword binary value, a browse token returned on a previous STARTBROWSE PROCESS command.

PROCESS(data-area)
returns the 36-character name of the next process.

Conditions

83 END
RESP2 values:

2 There are no more resource definitions of this type.

21 ILLOGIC
RESP2 values:

1 The value specified in the BROWSETOKEN option matches a current browse token, but not one that is being used for a process browse.

17 IOERR
RESP2 values:

30 An input/output error has occurred on the repository file.

108 PROCESSERR
RESP2 values:

13 The request timed out. It may be that another task using this process-record has been prevented from ending.

112 TOKENERR
RESP2 values:

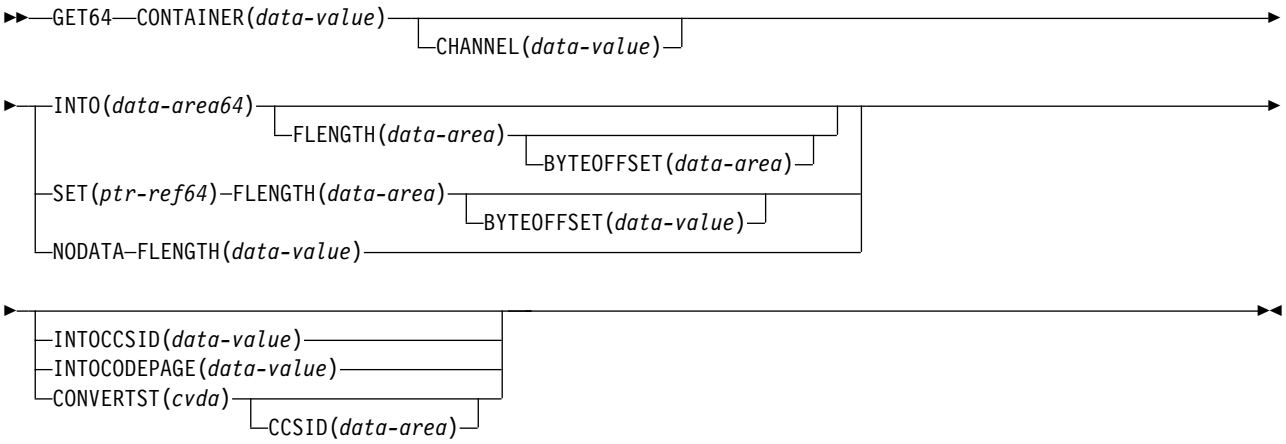
3 The browse token is not valid.

GET64 CONTAINER

Retrieve data from a named channel container into 64-bit storage. This command is for use only in non-Language Environment (LE) AMODE(64) assembler language application programs. CICS business transaction services (BTS) containers are not supported.

See Assembler language programming restrictions and requirements.

GET64 CONTAINER



Conditions: CCSIDERR, CHANNELERR, CODEPAGEERR, CONTAINERERR, INVREQ, LENGERR

This command is threadsafe.

Description

GET64 CONTAINER reads the data associated with a specified channel container into 64-bit storage.

The container that holds the data is identified by name and by the channel for which it is a container: the channel that "owns" it. The channel that owns the container can be identified explicitly, by specifying the CHANNEL option, or implicitly, by omitting the CHANNEL option. When this option is omitted, the current channel is implied.

Options

BYTEOFFSET(*data-value*)
Specifies the offset in bytes where the data returned starts. For CHAR containers, the BYTEOFFSET value is used as an offset into the data in the requested code page. If you use a code page with multibyte characters, depending on the BYTEOFFSET value you specify, the data returned might have partial characters at the beginning, end, or both. In this situation, your application program must be able to handle and interpret the data returned. If the value specified is less than zero, zero is assumed.

CCSID(*data-area*)
Returns a fullword that contains the Coded Character Set Identifier (CCSID) of the data returned by the CONVERTST(NOCONVERT) option. You can use this

option to retrieve containers with a DATATYPE of CHAR, without converting the data. If a DATATYPE of BIT is specified for the container, this value is zero.

CHANNEL(*data-value*)
Specifies the name (1 - 16 characters) of the channel that owns the container. You can specify the channel name DFHTRANSACTION to use the transaction channel.

CONTAINER(*data-value*)
Specifies the name (1 - 16 characters) of the container that holds the data to retrieve.

CONVERTST(*cvda*)
Specifies the required data conversion status.

NOCONVERT
The container data is retrieved without being converted. If you used the **WEB RECEIVE** command to store the HTTP body in a container, and you need to retrieve the body unconverted from that container, you must use the NOCONVERT option.

FLENGTH(*data-area*)
As an input field, FLENGTH specifies, as a fullword binary value, the length of the data to be read. As an output field, FLENGTH returns the length of the data in the container. FLENGTH is an input or an output field depending on which of the BYTEOFFSET, INTO, SET, or NODATA options you specify.

BYTEOFFSET option specified
FLENGTH is both an input and an output field.

On **input**, FLENGTH specifies the maximum length of the data that the program accepts. The data returned begins at the offset specified by the BYTEOFFSET value. If the value specified is less than zero, zero is assumed.

On **output** (that is, on completion of the retrieval operation) CICS sets the data area to the length of the data returned. The maximum length returned is equal to the length of the data in the container minus the BYTEOFFSET value.

INTO option specified
FLENGTH is both an input and an output field.

On **input**, FLENGTH specifies the maximum length of the data that the program accepts. If the value specified is less than zero, zero is assumed. If the length of the data exceeds the value specified, the data is truncated to that value and the LENGERR condition occurs. If the length of the data is less than the specified value, the data is copied but no padding is performed.

You do not need to specify FLENGTH if the length can be generated by the compiler from the INTO variable. If you specify both INTO and FLENGTH, FLENGTH specifies the maximum length of the data that the program accepts.

On **output** (that is, on completion of the retrieval operation) CICS sets the data area, if specified, to the actual length of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the data after conversion.

SET or NODATA option specified
FLENGTH is an output field only. It must be present and must be specified as a data-area.

On completion of the retrieval operation, the data area is set to the actual length of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the data after conversion.

INTO(*data-area64*)
Specifies the 64-bit data area into which the retrieved data is placed. *data-area64* refers to an area that is referenced by a 64-bit pointer and that can be in 64-bit (above-the-bar) storage.

INTOCCSID(*data-value*)
Specifies the Coded Character Set Identifier (CCSID) into which the character data in the container is converted, as a fullword binary number. If you prefer to specify an IANA name for the code page, or if you prefer to specify the CCSID as alphanumeric characters, use the INTOCODEPAGE option instead.

For CICS Transaction Server for z/OSapplications, the CCSID is typically an EBCDIC CCSID. However, it is possible to specify an ASCII CCSID if, for example, you want to retrieve ASCII data without it being automatically converted to EBCDIC.

If INTOCCSID and INTOCODEPAGE are not specified, the value for conversion defaults to the CCSID of the region. The default CCSID of the region is specified on the **LOCALCCSID** system initialization parameter.

Only character data can be converted, and only then if a DATATYPE of CHAR was specified on the **PUT CONTAINER** or **PUT64 CONTAINER** command used to place the data in the container. A DATATYPE of CHAR is implied if FROMCCSID or FROMCODEPAGE is specified on the **PUT CONTAINER** or **PUT64 CONTAINER** command.

For more information about data conversion with channels, see Data conversion with channels.

For an explanation of CCSIDs, see Preparing for code page conversion with channels.

INTOCODEPAGE(*data-value*)
Specifies an IANA-registered alphanumeric charset name or a Coded Character Set Identifier (CCSID) for the code page into which the character data in the container is converted, using up to 40 alphanumeric characters, including appropriate punctuation. Use this option instead of the CCSID option if you prefer to use an IANA-registered charset name, as specified in the Content-Type header for an HTTP request. CICS converts the IANA name into a CCSID, and the subsequent data conversion process is identical. Also use this option if you prefer to specify the CCSID in alphanumeric characters, rather than as a fullword binary number.

Where an IANA name exists for a code page and CICS supports its use, the name is listed with the CCSID. For more information, see Preparing for code page conversion with channels.

NODATA
Specifies that no data is retrieved. Use this option to discover the length of the data in the container (returned in FLENGTH).

The length of character data might change if data conversion takes place. Therefore, if character data is to be converted into any CCSID *other than that of*

this region, when you specify NODATA you should also specify INTOCCSID. This ensures that the correct length of the converted data is returned in FLENGTH.

SET(*ptr-ref64*)
Specifies a 64-bit pointer reference in which the 64-bit address of the retrieved data is returned. This pointer reference is always to 64-bit (above-the-bar) storage.

CICS maintains the data area until any of the following occurs:

- A subsequent **GET CONTAINER** or **GET64 CONTAINER** command with the SET option, for the same container in the same channel, is issued by any program that can access this storage.
- The container is deleted by a **DELETE CONTAINER** command.
- The container is moved by a **MOVE CONTAINER** command.
- The channel goes out of program scope.
- The channel, and the containers that are in it, are deleted by a **DELETE CHANNEL** command.

Beware of linking to other programs that might issue one of these commands.

Do not issue a **FREEMAIN64** command to release this storage.

If your application needs to keep the data, it should move it into its own storage.

Conditions

123 CCSIDERR
RESP2 values:

1 The CCSID specified on the INTOCCSID option is outside the range of valid CCSID values.

2 The CCSID specified on the INTOCCSID option and the CCSID of the container are an unsupported combination. (The CCSID of the container is the value that was specified using either FROMCODEPAGE or FROMCCSID, or defaulted, when the container was built.)

3 The data was created with a data type of BIT. Code page conversion is not possible. The data was returned without any code page conversion.

4 One or more characters could not be converted. The character has been replaced by a blank in the converted data.

5 There was an internal error in the code page conversion of a container.

122 CHANNELERR
RESP2 values:

2 The channel specified on the CHANNEL option could not be found.

125 CODEPAGEERR
RESP2 values:

1 The code page specified on the INTOCODEPAGE option is not supported.

2 The code page specified on the INTOCODEPAGE option and the code page of the channel are an unsupported combination.

3 The data was created with a data-type of BIT. Code page conversion is not possible. The data was returned without any code page conversion.

4 One or more characters could not be converted. The character has been replaced by a blank in the converted data.

5 There was an internal error in the code page conversion of a container.

110 CONTAINERERR
RESP2 values:

10 The container named on the CONTAINER option could not be found.

16 INVREQ
RESP2 values:

2 The INTOCCSID option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one.) INTOCCSID is valid only on GET64 CONTAINER commands that specify (explicitly or implicitly) a channel.

4 The CHANNEL option was not specified and there is no current channel (because the program that issued the command was not passed one).

5 The CONVERTST cvda value is invalid.

22 LENGERR
RESP2 values:

11 The length of the program area is shorter than the length of the data in the container. When the area is smaller, the data is truncated to fit into it.

12 The offset is greater than, or equal to, the length of the container.

HANDLE ABEND

Handle an abnormal termination exit.

HANDLE ABEND



Conditions: NOTAUTH, PGMIDERR (PROGRAM only)

This command is threadsafe.

Description

Use the **HANDLE ABEND** command to activate, cancel, or reactivate an exit for abnormal termination processing. You can suspend the command by using the **PUSH HANDLE** and **POP HANDLE** commands. See Using PUSH HANDLE and POP HANDLE commands.

When a task terminates abnormally, CICS searches for an active abend exit, starting at the logical level of the application program in which the abend occurred, and proceeding to successively higher levels. The first active abend exit found, if any, is given control.

The **HANDLE ABEND** command cannot intercept abends that are issued with the CANCEL option. Some internal abends generated by CICS are issued with the CANCEL option, for example the ASPx or APSJ abend codes.

When the label specified in a HANDLE ABEND LABEL command receives control, the registers are set as follows:

COBOL
Control returns to the **HANDLE ABEND** command with the registers restored. COBOL GO TO statement is then executed.

Assembler
R15: Abend label. R0-14: Contents at the point when the last EXEC CICS command was issued at the same program level as the **HANDLE ABEND** command was issued.

If LABEL is specified, the addressing mode and execution key used are those of the program that issued the HANDLE ABEND command.

If PROGRAM is specified, the addressing mode is defined by the way the program is link-edited and the execution key is specified by the EXECKEY option on the resource definition of the program.

If a COMMAREA has been established, it will be passed to the specified PROGRAM. Where more than one application program was involved in the task, the COMMAREA that is passed to the abend exit is the COMMAREA of the program that issued the HANDLE ABEND command. This might not be the COMMAREA of the program in which the abend occurred.

If a current channel exists, it will be accessible from the specified PROGRAM.

Options

CANCEL
Specifies that a previously established exit at the logical level of the application program in control is canceled. This option is the default.

LABEL(*label***)**
Specifies the program label to which control branches if abnormal termination occurs.

You cannot use this option for AMODE(64) assembler language, C, C++, or PL/I application programs.

PROGRAM(*name***)**
Specifies the name of the program that control is passed to if the task is terminated abnormally. If the abend condition is raised and the specified program is not already defined, this program is autoinstalled.

The program named in this option should always terminate with an abend, except when handling abends generated as a result of application program logic.

RESET
Specifies that an exit canceled by a HANDLE ABEND CANCEL command, or by CICS, is reactivated.

This option is usually issued by an abnormal termination exit routine.

Conditions

70 NOTAUTH
Occurs when a resource security check has failed on PROGRAM(name).

Default action: terminate the task abnormally.

27 PGMIDERR
RESP2 values:

1 The program has no installed resource definition and autoinstall for programs is not active.

2 The program is disabled.

9 The installed program resource definition is for a remote program.

Default action: terminate the task abnormally.

Examples

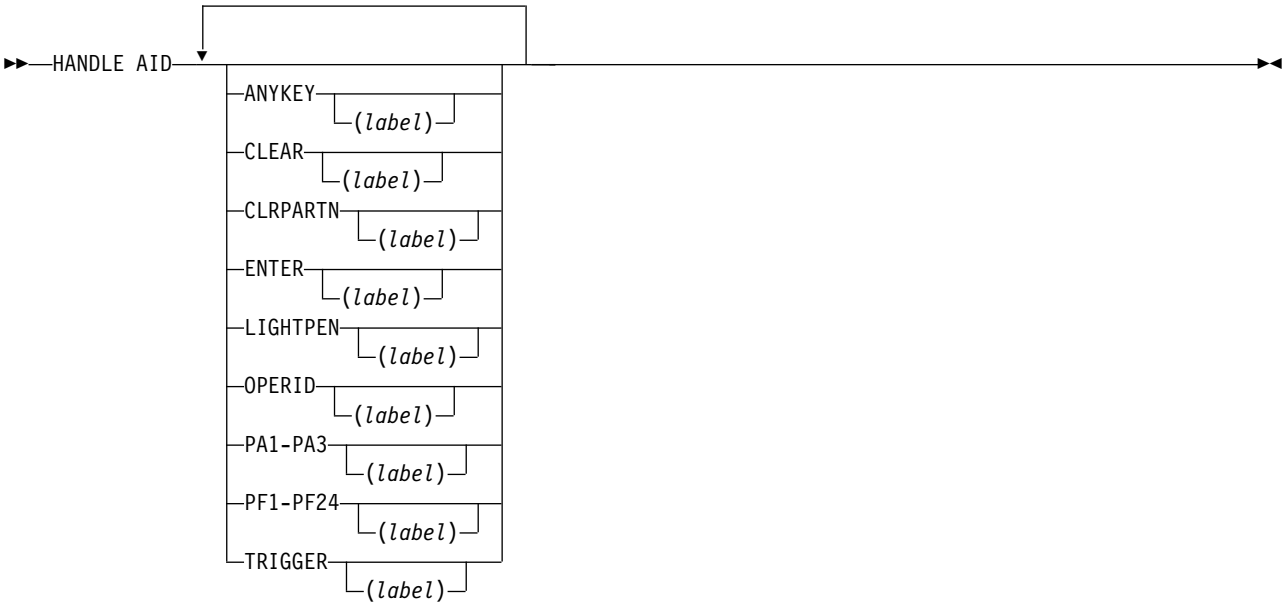
The following example shows how to establish a program as an exit:

```
EXEC CICS HANDLE ABEND
     PROGRAM('EXITPGM')
```

HANDLE AID

Handle attention identifiers (AIDs).

HANDLE AID



Condition: INVREQ

This command is threadsafe.

Description

You can use the HANDLE AID command to specify the label to which control is to be passed when an AID is received from a display device. Control is passed after the input command is completed; that is, after any data received in addition to the AID has been passed to the application program.

Restriction: This command is supported only in COBOL, PL/I, and assembler language applications (but not AMODE(64) assembler language applications). It is not supported in all other supported high level languages.

To ignore an AID, issue a HANDLE AID command that specifies the associated option **without** a label. This deactivates the effect of that option in any previously-issued HANDLE AID command.

If no HANDLE AID commands are in effect, that is, none have been issued or all have been canceled, control returns to the application program at the instruction immediately following the input command. Look in EIBAID to determine which key was pressed.

You can specify the following options:

- ANYKEY (any PA key, any PF key, or the CLEAR key, but not ENTER)
- CLEAR (for the key of that name)
- CLRPARTN (for the key of that name)
- ENTER (for the key of that name)
- LIGHTPEN (for a light-pen attention)

- OPERID (for the operator identification card reader, the magnetic slot reader (MSR), or the extended MSR (MSRE)
- PA1, PA2, or PA3 (any of the program access keys)
- PF1 through PF24 (any of the function keys)
- TRIGGER (for a trigger field attention)

You cannot include more than 16 options in the same command.

If a task is initiated from a terminal by means of an AID, the first RECEIVE command in the task does not read from the terminal but copies only the input buffer (even if the length of the data is zero) so that control can be passed by means of a HANDLE AID command for that AID.

For the standard attention identifier list (DFHAID), and the standard attribute and printer control character list (DFHBMSCA), see Chapter 9, "BMS-related constants," on page 849.

The label receives control in the same execution key as the execution key that the program was running in when the HANDLE AID command was issued.

A print key specified by the system PRINT initialization parameter takes precedence over a HANDLE AID command.

Conditions

16 INVREQ

RESP2 values:

200 The command was issued by a distributed program link server application.

Default action: terminate the task abnormally.

Example

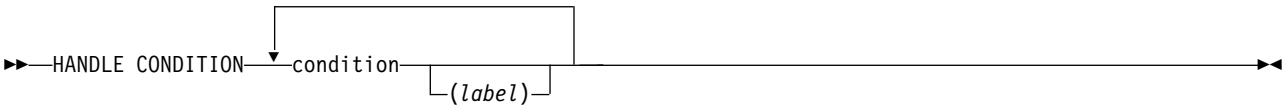
The following example shows a HANDLE AID command that specifies one label for the PA1 key, and a second label for CLEAR, PA2, PA3, and all the function keys except PF10. If a PF10 AID is received, or ENTER is pressed, control returns to the application program at the instruction immediately following the input command.

```
EXEC CICS HANDLE AID PA1(LAB1)
     ANYKEY(LAB2) PF10
```

HANDLE CONDITION

Handle conditions.

HANDLE CONDITION



This command is threadsafe.

Description

Use the **HANDLE CONDITION** command to specify the label to which control is to be passed if a condition occurs. You must include the name of the condition and, optionally, a label to which control is to be passed if the condition occurs.

Restriction: This command is supported only in COBOL, PL/I, and assembler language applications (but not AMODE(64) assembler language applications). It is not supported in all other supported high level languages.

If you omit the label parameter, any **HANDLE CONDITION** command for the condition is deactivated, and the default action is taken if the condition occurs. This is independent of the setting of the generalized ERROR condition.

You must ensure that the **HANDLE CONDITION** command is executed before the command that might result in the associated condition.

You cannot include more than 16 conditions in the same command. The conditions must be separated by at least one space. You must specify any additional conditions in further **HANDLE CONDITION** commands.

If a condition occurs that is not specified in a **HANDLE CONDITION** or **IGNORE CONDITION** command, the default action is taken. However, if the default action for such a condition terminates the task abnormally, and the condition ERROR has been specified, the action for ERROR is taken.

The label receives control in the same execution key as the execution key that the program was running in when the **HANDLE CONDITION** command was issued.

Scope

The **HANDLE CONDITION** command for a given condition applies only to the program in which it is specified. The **HANDLE CONDITION** command remains active while the program is being executed, or until one of the following situations occurs:

- An **IGNORE CONDITION** command for the same condition is encountered. The **HANDLE CONDITION** command is overridden.
- Another **HANDLE CONDITION** command for the same condition is encountered. The new command overrides the previous one.
- A **LINK** command is executed to call another CICS program. The **HANDLE CONDITION** options are not inherited by the linked-to program.

The **HANDLE CONDITION** command is temporarily deactivated by the NOHANDLE or RESP option on a command.

Language considerations

In an assembler language application program, when a branch to a label is caused by a condition, the registers in the application program are restored to their values in the program at the point where the command that caused the condition is issued.

In a PL/I application program, a branch to a label in an inactive procedure or in an inactive begin block, caused by a condition, produces unpredictable results.

Options

condition(*label***)**
Specifies the name of the condition. *label* specifies the location in the program to be branched to if the condition occurs.

For more information about the conditions, see EIB fields.

Examples

The following example shows how to handle conditions such as DUPREC and LENGERR that can occur when you use a WRITE command to add a record to a data set. DUPREC is handled as a special case. Default action is taken for LENGERR (that is, the task is terminated abnormally). All other conditions are handled by the error routine ERRHANDL.

```
EXEC CICS HANDLE CONDITION
          ERROR(ERRHANDL)
          DUPREC(DUPRTN) LENGERR
```

IGNORE CONDITION

Ignore conditions.

IGNORE CONDITION



This command is threadsafe.

Description

Use the **IGNORE CONDITION** command to specify that no action is taken if a condition occurs (that is, control returns to the instruction following the command that failed to execute, and the EIB is set). Execution of a command might result in several conditions being raised. CICS checks these conditions in a predetermined order. Only the first one that is not ignored (by your **IGNORE CONDITION** command) is passed to your application program.

Restriction: This command is supported only in COBOL, PL/I, and assembler language applications (but not AMODE(64) assembler language applications). It is not supported in all other supported high level languages.

For information about the conditions, see EIB fields.

The **IGNORE CONDITION** command for a given condition applies only to the program in which it is specified, and it remains active while the program is being executed, or until a **HANDLE CONDITION** command for the same condition is encountered, in which case the **IGNORE CONDITION** command is overridden.

You cannot include more than sixteen conditions in the same command. The conditions must be separated by at least one space. You can specify additional conditions in further IGNORE CONDITION commands.

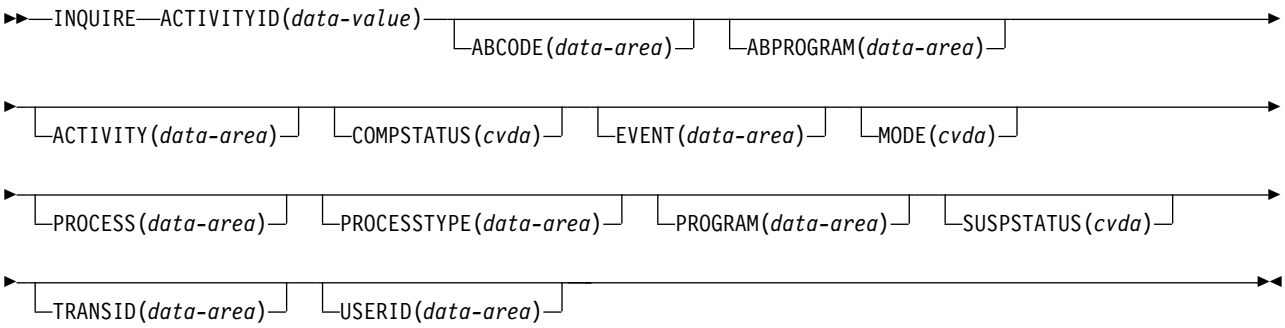
Options

condition
Specifies the name of the condition to be ignored.

INQUIRE ACTIVITYID

Retrieve the attributes of a BTS activity.

INQUIRE ACTIVITYID



Conditions: ACTIVITYERR, NOTAUTH

Description

INQUIRE ACTIVITYID returns the attributes of a specified BTS activity.

You can use this command to get details of an activity whose identifier has been retrieved during a browse operation.

Options

ABCODE(data-area)
returns, if the activity terminated abnormally, the 4-character abend code.

ABPROGRAM(data-area)
returns, if the activity terminated abnormally, the 8-character name of the program that was in control at the time of the abend.

ACTIVITY(data-area)
returns the 16-character name of the activity being queried.

ACTIVITYID(data-value)
specifies the identifier (1–52 characters) of the activity to be queried. (Typically, the activity identifier will have been retrieved by a GETNEXT ACTIVITY command, during an activity browse.)

COMPSTATUS(cvda)
indicates the completion status of the activity. CVDA values are:

ABEND
The program that implements the activity abended. Any children of the activity have been canceled.

FORCED
The activity was forced to complete—for example, it was canceled with a CANCEL ACTIVITY command.

INCOMPLETE
The named activity is incomplete. This could mean:

- That it has not yet been run
- That it has returned from one or more activations but needs to be reattached in order to complete all its processing steps
- That it is currently active.

NORMAL
The named activity completed successfully.

EVENT(data-area)
returns the 16-character name of the completion event that is sent to the requestor of this activity when the activity completes asynchronously with the requestor.

MODE(cvda)
indicates the current state (mode) of the activity. CVDA values are:

ACTIVE
An activation of the activity is running.

CANCELLING
CICS is waiting to cancel the activity. A CANCEL ACTIVITY command has been issued, but CICS cannot cancel the activity immediately because one or more of the activity's children are inaccessible.

No further operations on the activity are permitted until it has been canceled.

COMPLETE
The activity has completed, either successfully or unsuccessfully. The value returned on the COMPSTATUS option tells you how it completed.

DORMANT
The activity is waiting for an event to fire its next activation.

INITIAL
No RUN or LINK command has yet been issued against the activity; or the activity has been reset by means of a RESET ACTIVITY command.

PROCESS(data-area)
returns the 36-character name of the process to which this activity belongs.

PROCESSTYPE(data-area)
returns the 8-character name of the process-type to which the process that contains this activity belongs.

PROGRAM(data-area)
returns the 8-character name of the program that executes when this activity is run.

SUSPSTATUS(cvda)
indicates whether the activity is currently suspended. CVDA values are:

SUSPENDED
The activity is currently suspended. If a reattachment event occurs, it will not be reactivated.

NOTSUSPENDED
The activity is not currently suspended. If a reattachment event occurs, it will be reactivated.

TRANSID(data-area)
returns the 4-character transaction identifier under which this activity runs.

USERID(data-area)
returns the 8-character identifier of the user under whose authority this activity runs.

Conditions

109 ACTIVITYERR
RESP2 values:

1 The activity identifier specified on the ACTIVITYID option does not relate to any activity that is within the scope of this task.

19 The request timed out. It may be that another task using this activity-record has been prevented from ending.

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

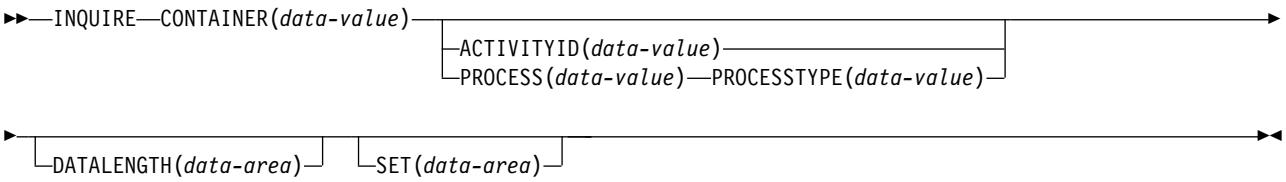
70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access this resource in the way requested.

INQUIRE CONTAINER

Retrieve the attributes of a BTS data-container.

INQUIRE CONTAINER



Conditions: ACTIVITYERR, CONTAINERERR, IOERR, NOTAUTH, PROCESSERR

Description

INQUIRE CONTAINER returns a pointer to the contents of a named BTS data-container, plus the length of the data.

To inquire upon a container associated with the current activity, omit the ACTIVITYID and PROCESS options.

To inquire upon a container associated with another activity, specify the ACTIVITYID option. (The activity identifier specified on the ACTIVITYID option may, for example, have been returned on a GETNEXT ACTIVITY command during a browse operation.)

To inquire upon a process container (including one associated with the *current* process), specify the PROCESS and PROCESSTYPE options.

Note:

1. Inquiring on a container of the current activity returns details of the in-storage version, rather than the committed version on the repository. This means that it's possible to see:
   - Containers that are not yet on the repository
   - Container contents that differ from those on the repository.
2. Inquiring on a container not owned by the current activity returns details of the committed version on the repository. However, the read of the repository record is “dirty”—the record is not locked. So, if the record is being updated by another task, it's possible for the returned data to be unreliable.

Options

ACTIVITYID(data-value)
specifies the identifier (1–52 characters) of the activity which the data-container is associated with.

If both this and the process options are omitted, the current activity is assumed.

CONTAINER(data-value)
specifies the name (1–16 characters) of the data-container being inquired upon.

DATALENGTH(data-area)
returns the fullword length of the data contained in the named data-container.

PROCESS(data-value)
specifies the name (1–36 characters) of the process which the data-container is associated with.

If both this and the ACTIVITYID option are omitted, the current activity is assumed.

PROCESSTYPE(data-value)
specifies the process-type (1–8 characters) of the process named in the PROCESS option.

SET(data-area)
returns a pointer to the contents of the data-container.

Conditions

109 ACTIVITYERR
RESP2 values:

2 The activity indicated by the ACTIVITYID option could not be found.

3 Because neither the ACTIVITYID nor the PROCESS options were specified, an inquiry on the current activity was implied—but there is no current activity associated with the request.

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

110 CONTAINERERR
RESP2 values:

1 The container specified on the CONTAINER option could not be found.

17 IOERR
RESP2 values:

30 An input/output error has occurred on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access this resource in the way requested.

108 PROCESSERR
RESP2 values:

2 The process-type specified on the PROCESSTYPE option could not be found.

4 The process specified on the PROCESS option could not be found.

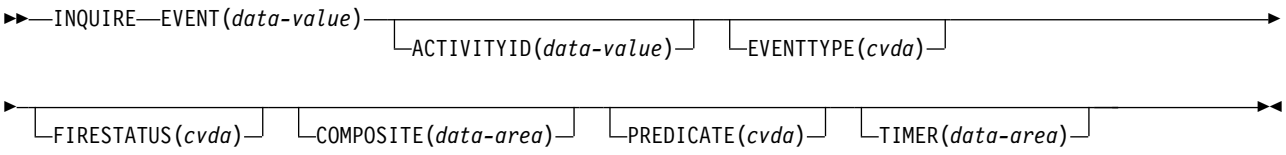
13 The request timed out. It may be that another task using this process-record has been prevented from ending.

33 The process specified on the PROCESS option has not yet been committed.

INQUIRE EVENT

Retrieve the attributes of a BTS event.

INQUIRE EVENT



Conditions: ACTIVITYERR, EVENTERR, INVREQ, IOERR, NOTAUTH

Description

INQUIRE EVENT returns the attributes of a named BTS event.

To inquire upon an event associated with the current activity, omit the ACTIVITYID option. To inquire upon an event associated with another activity, specify the ACTIVITYID option. (The activity identifier specified on the ACTIVITYID option may, for example, have been returned on a GETNEXT ACTIVITY command during a browse operation.)

Options

ACTIVITYID(data-value)
specifies the identifier (1–52 characters) of the activity which the event is associated with.

If this option is omitted, the current activity is assumed.

COMPOSITE(data-area)
returns, if the named event is a sub-event, the 16-character name of the composite event that it is part of.

EVENT(data-value)
specifies the name (1–16 characters) of the event being inquired upon.

EVENTTYPE(cvda)
indicates the type of the named event. CVDA values are:

ACTIVITY
Activity completion

COMPOSITE
Composite

INPUT
Input

SYSTEM
System

TIMER
Timer.

FIRESTATUS(cvda)
indicates the state of the named event. CVDA values are:

FIRED
The event has fired normally.

NOTFIRED
The event has not fired.

PREDICATE(cvda)
indicates, if the named event is composite, the Boolean operator applied to its predicate. CVDA values are:

AND The Boolean operator applied to the predicate is AND.

OR The Boolean operator applied to the predicate is OR.

TIMER(data-area)
returns, if the named event is a timer event, the 16-character name of the timer.

Conditions

109 ACTIVITYERR
RESP2 values:

3 The activity indicated by the ACTIVITYID option could not be found.

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

111 EVENTERR
RESP2 values:

1 The event specified on the EVENT option could not be found.

16 INVREQ
RESP2 values:

1 There is no current activity within the scope of this task.

17 IOERR
RESP2 values:

30 An input/output error has occurred on the repository file.

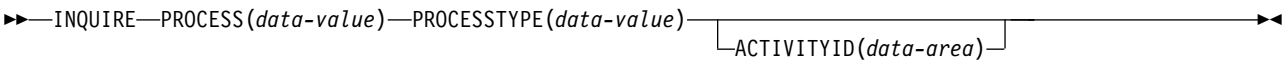
70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access this resource in the way requested.

INQUIRE PROCESS

Retrieve the attributes of a BTS process.

INQUIRE PROCESS



Conditions: ILLOGIC, NOTAUTH, PROCESSERR

Description

INQUIRE PROCESS returns the attributes of a named BTS process. It can be used, for example, to obtain the identifier of the root activity of a process, in order to start a browse of the root activity's child activities, containers, or events.

Options

ACTIVITYID(data-area)
returns the 52-character identifier of the root activity of the process that is being queried.

PROCESS(data-value)
specifies the name (1–36 characters) of the process to be queried.

PROCESSTYPE(data-value)
specifies the process-type (1–8 characters) of the process to be queried.

Conditions

21 ILLOGIC
RESP2 values:

1 A browse of this resource type is already in progress.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access this resource in the way requested.

108 PROCESSERR
RESP2 values:

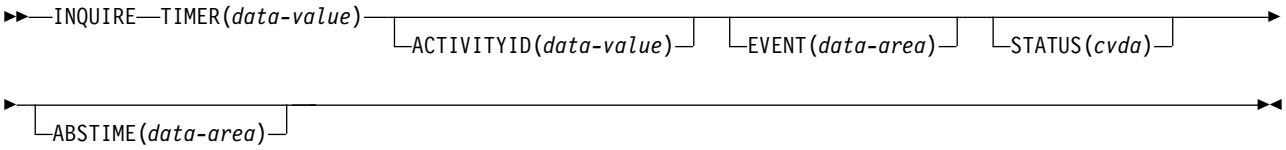
1 The process specified on the PROCESS option could not be found.

4 The process-type specified on the PROCESSTYPE option could not be found.

INQUIRE TIMER

Retrieve the attributes of a BTS timer.

INQUIRE TIMER



Conditions: ACTIVITYERR, INVREQ, IOERR, NOTAUTH, TIMERERR

Description

INQUIRE TIMER returns the attributes of a named BTS timer.

To inquire upon a timer associated with the current activity, omit the ACTIVITYID option. To inquire upon a timer associated with another activity, specify the

ACTIVITYID option. (The activity identifier specified on the ACTIVITYID option may, for example, have been returned on a GETNEXT ACTIVITY command during a browse operation.)

Options

ABSTIME(data-area)
returns, in packed decimal format, the time at which the timer will expire, expressed in milliseconds since 00:00 on 1 January 1900 (rounded to the nearest hundredth of a second).

You can use FORMATTIME to change the data into other familiar formats.

ACTIVITYID(data-value)
specifies the identifier (1–52 characters) of the activity which the timer is associated with.

If this option is omitted, the current activity is assumed.

EVENT(data-area)
returns the 16-character name of the event (if any) associated with the timer.

STATUS(cvda)
indicates the state of the timer. CVDA values are:

EXPIRED
The timer expired normally.

FORCED
Expiry of the timer was forced by means of a FORCE TIMER command.

UNEXPIRED
The timer has not yet expired.

TIMER(data-value)
specifies the name (1–16 characters) of the timer.

Conditions

109 ACTIVITYERR
RESP2 values:

3 The activity indicated by the ACTIVITYID option could not be found.

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

16 INVREQ
RESP2 values:

1 The command was issued outside the scope of a currently—active activity.

17 IOERR
RESP2 values:

30 An input/output error has occurred on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access this resource in the way requested.

115 TIMERERR
RESP2 values:

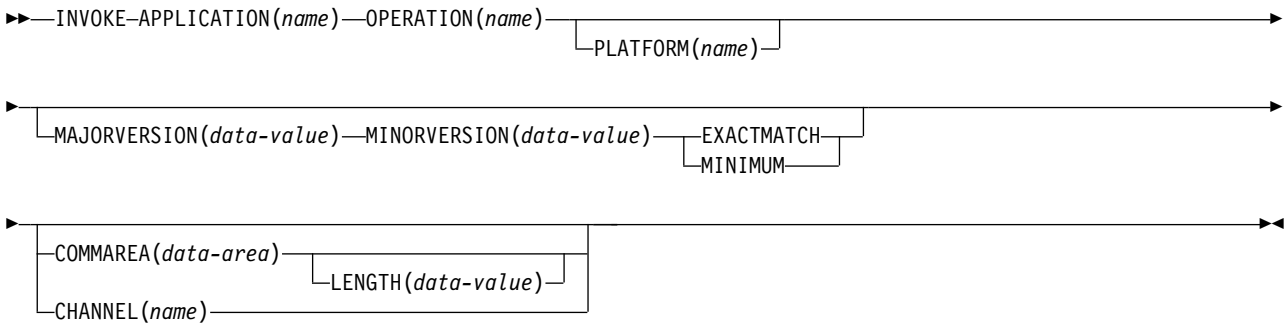
1 The timer specified on the TIMER option could not be found.

INVOKE APPLICATION

Invoke an application entry point program. **EXEC CICS INVOKE APPLICATION** allows invocation of an application by naming an operation that corresponds to one of its program entry points, without having to know the name of the application entry point program and regardless of whether the program is public or private.

The application must be in an AVAILABLE state. If multiple versions of the application are in an AVAILABLE state, the entry point programs of the highest version of the application are the ones that are public, and so can be linked to by using an **EXEC CICS LINK** command. The **INVOKE APPLICATION** command allows lower versions of the application to be invoked, by specifying a version as well as an operation, and this corresponds to a private entry point program.

INVOKE APPLICATION



Conditions: APPNOTFOUND, CHANNELERR, INVREQ, LENGERR, NOTAUTH, PGMIDERR

This command is threadsafe.

Description

The **INVOKE APPLICATION** command links to a local program, which is the entry point to the specified application for the specified operation. It passes control from an application program at one logical level to an application program at the next lower logical level.

This command operates in the current platform context. If the command does not specify a platform name, the current platform name is used. If there is no current platform, then the command fails with a response of APPNOTFOUND.

For more information, and examples of how you can use the **EXEC CICS INVOKE APPLICATION** command, see Invoking a multi-versioned application in Administering.

Options

APPLICATION(*name*)
Specifies the name (1 - 64 characters) of the application. The acceptable characters are: a-z A-Z 0-9 . _ # @ -

CHANNEL(*name*)
Specifies the name (1 - 16 characters) of a channel that is to be made available to the called program. The acceptable characters are: A-Z a-z 0-9 $ @ # / % & ? ! : | " = ¬ , ; < > . - _. Leading and embedded blank characters are not permitted. If the name supplied is fewer than 16 characters, it is padded with trailing blanks up to 16 characters. If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

Channel names are always in EBCDIC. The set of allowed characters for channel names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be transferred between regions, it is advisable to restrict the characters that are used to name them to A-Z a-z 0-9 & : =, ; < > . - and _.

You can specify the channel name DFHTRANSACTION to use a transaction channel. A transaction channel does not go out of scope when the link level changes: it is always accessible in the transaction. For more information, see Channels and containers.

The program that issues the INVOKE command can do one of the following:
- Have already created the channel by using one or more **PUT CONTAINER CHANNEL** or **PUT64 CONTAINER** commands.
- Specify its current channel, by name.
- Name a channel that does not currently exist. A new empty channel is created.

COMMAREA(*data-area*)
Specifies a communication area that is to be made available to the called program. In this option, the data area is passed, and you must give it the name DFHCOMMAREA in the receiving program. See Passing data to other programs .

EXACTMATCH
Specifies that an exact match on the application major version number and minor version number is required. If not found then an APPNOTFOUND condition is returned.

Note: There is no match criteria for micro version. The highest micro version is always used.

LENGTH(*data-value*)
Specifies a halfword binary value that is the length in bytes of the COMMAREA (communication area). This value must not exceed 24 KB if the COMMAREA is to be passed between any two CICS servers (for any combination of product, version, and release). This limit allows for the COMMAREA and space for headers.

Ensure that the value you specify matches the length of the data that is being passed in the COMMAREA. Do not specify 0 (zero) for LENGTH because the resulting behavior is unpredictable and the **EXEC CICS LINK** command might fail.

When you use a COMMAREA to pass data, the program that is linked to must verify that the EIBCALEN field in the EIB of the task matches what the

program expects. Discrepancies might result in storage violations or system failures. For more information, see COMMAREA.

MAJORVERSION(*data-value***)**
Specifies the major version number of the application as a fullword binary value.

If MAJORVERSION is specified, then MINORVERSION must also be specified. If no version is specified, then the highest major and minor version of the application is invoked.

MINIMUM
Specifies that the specified minor version number is the minimum that is required, but use a higher version if it is available. If multiple higher minor versions are available the highest is used. This applies to minor version numbers only. The major version number cannot be exceeded and must match exactly. If no higher minor version exists nor the minimum required minor version, then an APPNOTFOUND condition is returned.

Note: There is no match criteria for micro version. The highest micro version is always used.

MINORVERSION(*data-value***)**
Specifies the minor version number of the application as a fullword binary value.

If MINORVERSION is specified, then MAJORVERSION must also be specified. If no version is specified, then the highest major and minor version of the application is invoked.

The EXACTMATCH or MINIMUM keywords specify the matching criteria for the major and minor versions.

OPERATION(*name*)
Specifies the name (1 - 64 characters) of the application operation which the application entry point program implements. The acceptable characters are: a-z A-Z 0-9 . _ # @ -

PLATFORM(*name*)
Specifies the name (1 - 64 characters) of the platform on which the application is installed. The acceptable characters are: a-z A-Z 0-9 . _ # @ -

If no platform name is specified, the current platform name is used. If there is no current platform, then the command fails with a response of APPNOTFOUND.

Conditions

127 APPNOTFOUND
RESP2 values:

1 The EXACTMATCH keyword is specified and the required version of the application cannot be found.

2 The MINIMUM keyword is specified and no minimum level or higher microversion of the application can be found.

3 No version was specified. No application can be found.

Default action: end the task abnormally.

An application might not be found for a number of reasons:

- The application is not in an AVAILABLE state.

- The named operation does not correspond to an entry point program for the application.
- The application is not installed on this platform.
- The CICS region is not part of the platform name specified.

122 CHANNELERR
RESP2 values:

1 The name that is specified on the CHANNEL option contains an invalid character or combination of characters.

Default action: end the task abnormally.

16 INVREQ
RESP2 values:

1 No Platform name is specified and there is no current platform.

2 The application entry point program is a Java™ program but the user class cannot be found.

3 The application entry point program is a Java program but the JVMSERVER cannot be found.

4 The application entry point program is a Java program but the JVMSERVER resource is not enabled.

Default action: end the task abnormally.

22 LENGERR
RESP 2 values:

11 The COMMAREA length is less than 0 or greater than the permitted length.

26 The COMMAREA address is zero, but the COMMAREA length is nonzero.

Default action: end the task abnormally.

70 NOTAUTH
RESP2 values:

101 A resource security check has failed on the name of the application entry point program that implements the operation for the specified application.

Default action: end the task abnormally.

27 PGMIDERR
RESP2 values:

1 The application entry point program is disabled.

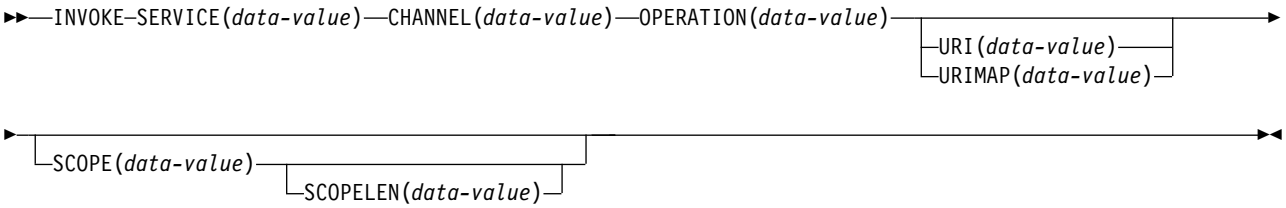
2 The application entry point program could not be loaded

Default action: end the task abnormally.

INVOKE SERVICE

Call a service from a CICS application. The command specifies the name of a service or the CICS resource, such as a WEBSERVICE resource, that contains information about the service to be called.

INVOKE SERVICE



Conditions: INVREQ, LENGERR, NOTFND, TIMEDOUT

This command is threadsafe.

Description

Use the **INVOKE SERVICE** command in CICS applications to call a service; for example, the application can act as a web service requester and call an XML-based service, or the application can call another CICS application that is acting as a channel-based service. For more information about these two types of services, see Creating business services from CICS applications.

Use this command for all new web service applications, rather than the **INVOKE WEBSERVICE** command, which is a synonym. If you use this command for web services, you must supply specific containers to CICS as input. For more information on writing a web service application, see Creating a SOAP web service.

When you invoke the service, you can specify a URIMAP resource that contains the information about the URI of the service. You can specify this information directly on the INVOKE SERVICE command instead of using a URIMAP resource. However, using a URIMAP resource has the following advantages:

- System administrators can manage any changes to the endpoint of the connection, so you do not need to recompile your applications if the URI of a service provider changes.
- You can choose to make CICS keep the connections that were opened with the URIMAP resource open after use, and place them in a pool for reuse by the application for subsequent requests, or by another application that calls the same service. Connection pooling is only available when you specify a URIMAP resource that has the SOCKETCLOSE attribute set. For more information about the performance benefits of connection pooling, see Connection pooling for HTTP client performance.

The **INVOKE SERVICE** command drives the XWBOPEN user exit, which can make the connection to the server go through a proxy server, if required.

Options

CHANNEL(*data-value*)
Specifies the name of the channel used to pass the containers that hold the data mapped by the application data structure. On return, the same channel holds the response from the web service, again mapped by the application data structure. The name of the channel can be up to 16 characters. If *name* is a variable, and it contains a name that is less than 16 characters, then the variable must be padded with trailing blanks. You can specify the channel name DFHTRANSACTION to use the transaction channel.

OPERATION(*data-value*)
Specifies a data area containing the name of the operation that is to be invoked. The name of the operation is contained in the WSDL for the target web service. The data area must be 255 characters long; if the operation name is less than 255 characters, then the data area must be padded with trailing blanks.

SERVICE(*data-value*)
Specifies the name of the service:

- If you want to invoke a web service, specify the name of the WEBSERVICE resource that defines the web service. The WEBSERVICE resource specifies the location of the web service description and the web service binding file that CICS uses when it communicates with the web service. The name of the WEBSERVICE resource can be up to 32 characters. The value must be padded with trailing blanks if there are less than 32 characters.
- If you want to invoke a channel-based service, specify the name of the service. The format of the service is a URI. The name can be up to 32 characters. The value must be padded with trailing blanks if there are less than 32 characters.

SCOPE(*data-value*)
Acts as a qualifying prefix for the service name. Use SCOPE if you are using the Service Component Architecture (SCA) channel-based services and require an additional prefix to identify the service.

Note: SCOPE only applies to channel-based services.

If you are writing a COBOL program that is translated with the COBOL3 translator option, the length of the data value cannot exceed 160 bytes. If you are using the COBOL2 translator option, you must use a data area instead of a data value.

SCOPELEN(*data-value*)
A fullword binary value that specifies the length of the scope that prefixes the service name.

URI(*data-value*)
Specifies a data area containing the URI of the service to be invoked. If specified, this option supersedes any URI specified in the WEBSERVICE resource definition. If you omit both this option and the URIMAP option, the WEBSERVICE binding file associated with the resource definition must include either a provider URI or a provider application name. The data area must be 255 characters long; if the URI is less than 255 characters, the data area must be padded with trailing blanks. For information about the format of URIs, see The components of a URL.

Do not specify this option for web services that use WS-Addressing.

Do not specify this option if you use connection pooling; use the URIMAP option instead to specify an appropriate URIMAP resource to enable connection pooling.

URIMAP(*data-value*)
Specifies the name of a URIMAP resource that CICS uses to derive the URI value. Use a URIMAP resource if you want to enable connection pooling, where CICS keeps the client HTTP connection open for this application or another application to reuse. If specified, this option supersedes any URI specified in the WEBSERVICE resource definition. If you omit both this option

and the URI option, the WEBSERVICE binding file associated with the resource definition must include either a provider URI or a provider application name.

You must create the URIMAP resource for an HTTP client request with the attribute USAGE(CLIENT). For connection pooling, you must also set the SOCKETCLOSE attribute. The CICS web services assistant does not create the URIMAP resource, so you must define it yourself. For information about creating a URIMAP resource for a client request, see Creating a URIMAP resource for CICS as a HTTP client.

Do not specify this option for web services that use WS-Addressing.

Conditions

16 INVREQ

RESP2 values:

1 The name specified for the CHANNEL option contains an illegal character or combination of characters.

2 The name specified for the OPERATION option contains an illegal character or combination of characters.

3 The web service binding file associated with the WEBSERVICE is invalid.

4 The value specified for the URI contained an illegal character or combination of characters, or the specified host name could not be resolved.

5 The PIPELINE used by the WEBSERVICE is defined as a service requester pipeline but is invoked in a service provider or *vice versa*.

6 The invoked WEBSERVICE returned a SOAP fault. The description of the fault is available in its XML format in the container DFHWS-BODY.

Note: This condition is not raised for XML-ONLY web service invocations.

7 The URI option was not specified on the command, and the WEBSERVICE definition does not specify a URI or a program name.

8 The WEBSERVICE is not in service

9 A container does not have the correct DATATYPE. This may be the **DFHWS-DATA** container, or another container referenced in the application data. The **DFHWS-DATA** container and most other application data containers must be populated in BIT mode. Any containers that hold XML markup must be populated in CHAR mode.

10 The PIPELINE used by the WEBSERVICE is not enabled.

11 CICS could not link to the program specified in the WEBSERVICE definition.

12 The containers that the command expects were not on the correct channel.

13 An input error was detected either generating a SOAP request message or processing a SOAP response message. A DFHPI*xxxx* message is written to MSGUSR to document the problem in more detail. It is likely that the application data structure contains invalid data that cannot be converted to a SOAP request message. For more information, see the error message in the DFH-XML-ERRORMSG container.

14 A conversion error occurred when CICS attempted to convert between the application data structure and the SOAP message. Either the application data structure contains invalid data that cannot be converted to a SOAP request, or data in the SOAP response message cannot be converted into the application's data structure. Some possible causes of this condition are:

- A value in the SOAP response message is larger than the corresponding field in the application's data structure.
- When building the SOAP request, the web services binding file indicates that a data field contains packed decimal or zoned decimal data, and the contents of the field are invalid for this data type.

A DFHPI*xxxx* message is written to MSGUSR to document the problem in more detail. For more information, see the error message in the DFH-XML-ERRORMSG container.

15 An unhandled error has occurred in the pipeline. Information about the error is in container DFHERROR.

16 A locally optimized web service has abended. The underlying unit of work has been backed out.

17 A remote web service request did not return a response message.

18 The container **DFHWS-BODY** has not been populated by an application for an XML-ONLY WEBSERVICE.

19 A URI or a URIMAP has been specified, but this option is not allowed when the WEBSERVICE resource has a default WS-Addressing endpoint reference or the WS-Addressing context has been built using the **WSACONTEXT BUILD** API command.

20 The specified URIMAP does not have a valid scheme.

21 The specified URIMAP is not client mode.

22 The specified URIMAP is not enabled.

23 An unspecified transport or link failure occurred when attempting to use the pipeline. CICS issues a message to document the specific problem.

24 z/OS Connect pipeline was used.

41 The connection has been closed.

101 The container **DFHWS-BODY** does not have the correct DATATYPE. For this container, a DATATYPE of CHAR must be specified.

103 The container **DFHWS-BODY** contains no data.

104 Either the container **DFHREQUEST** or the container **DFHWS-BODY** is missing.

105 A fault was built within the service requester PIPELINE used by the WEBSERVICE, either while the request was being sent, or while the response was being processed. This condition could indicate that a header processing program has issued a fault.

106 Either the generated SOAP request message was not well formed, or the SOAP response message was not well formed. This condition could indicate that the XML parser returned a fatal error code.

107 Either the generated SOAP request message was not a valid SOAP message, or the SOAP response message was not a valid SOAP message.

109 109 HTTP redirect (301, 302, 303, or 307) response was returned. The Location header is available in the **DFHWS-LOCATION** container.

22 LENGERR
RESP 2 values:

1 Either the SCOPELEN option was not specified or it was not a valid value.

13 NOTFND
RESP2 values:

1 The web service binding file associated with the WEBSERVICE specifies the name of a SOAP message parsing program supplied by another product, but the parsing program could not be found.

2 The specified CHANNEL could not be located.

3 The specified OPERATION was not in the web service binding file.

4 The specified WEBSERVICE could not be located.

5 A CONTAINER specified in the web service binding file could not be located.

6 The specified URIMAP could not be located.

124 TIMEDOUT
RESP2 values:

1 An expected timeout has occurred. When the message exchange pattern specifies an optional error response, and an error response is not returned from the remote web service, the timeout is acceptable.

2 An unexpected timeout has occurred. A response was expected from the remote web service, but none was received.

62 An unexpected timeout has occurred on socket receive.

INVOKE WEBSERVICE

Call a service from a CICS application. This command is a synonym of the **INVOKE SERVICE** command and is provided for compatibility with existing web service requester applications. Use **INVOKE SERVICE** for any new web service applications.

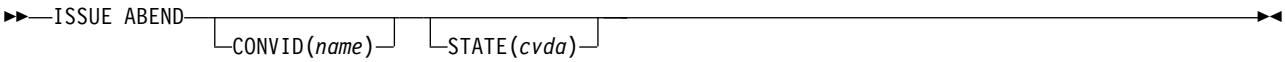
Description

For details of the **INVOKE SERVICE** command, see "INVOKE SERVICE" on page 281.

ISSUE ABEND

Abend the mapped conversation with an APPC partner.

ISSUE ABEND (APPC)



Conditions: INVREQ, NOTALLOC, TERMERR

Description

ISSUE ABEND abnormally ends the conversation. The partner transaction sees the TERMERR condition.

Options

CONVID(*name***)**
identifies the conversation to be abended. The 4-character name identifies either the symbolic identifier returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the symbolic identifier representing the principal facility (returned by a previously executed ASSIGN command).

For compatibility with earlier releases, SESSION is accepted as a synonym for CONVID. New programs should use CONVID.

If both CONVID and SESSION are omitted, the principal facility is assumed.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

also occurs (RESP2 not set) in any of the following situations:

- ISSUE ABEND is used on any conversation other than an EXEC CICS APPC mapped conversation.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the specified CONVID value relates to a conversation that is not owned by the application.

Default action: terminate the task abnormally.

81 TERMERR
occurs for a session-related error. Any action on that conversation other than a FREE command causes an ATCV abend.

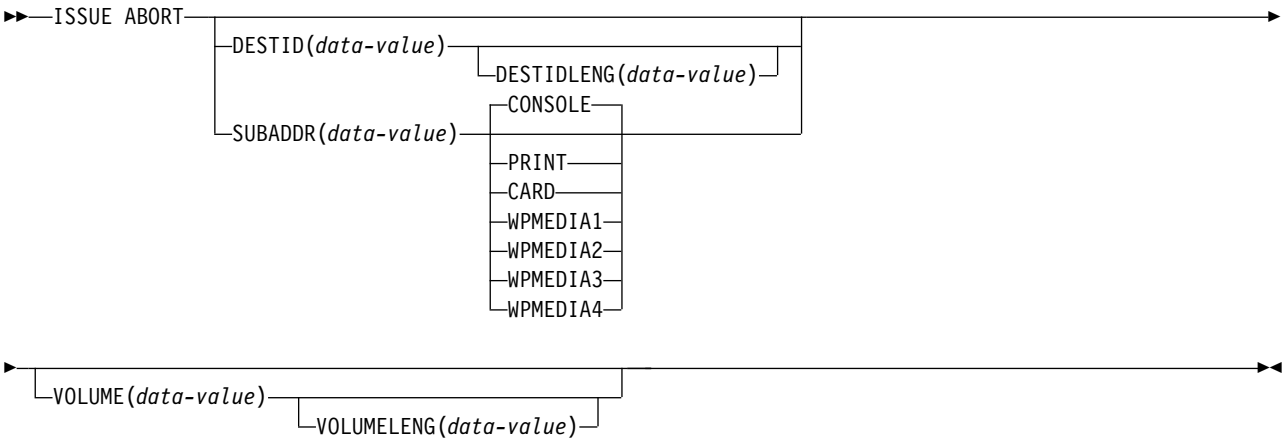
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE ABORT

End processing of a data set abnormally.

ISSUE ABORT



Conditions: FUNCERR, INVREQ, SELNERR, UNEXPIN

Description

ISSUE ABORT ends communication with a data set in an outboard controller, or the selected medium, abnormally. The data set specified in the DESTID option is deselected abnormally. The options CONSOLE, PRINT, CARD, and WPMEDIA1–4 are alternatives to DESTID and DESTIDLENG.

Options

CARD
specifies that the output medium is a card reader or card punch device. This option is not valid with DESTID and DESTIDLENG.

CONSOLE
specifies that the output medium is that provided for messages to the operator. This option is not valid with DESTID and DESTIDLENG. This refers to a programmable subsystem such as the IBM® 3790 data communication system. It does not refer to a CICS or system console.

DESTID(*data-value***)**
specifies the name (1–8 characters) of the data set in the outboard destination.

DESTIDLENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the DESTID option.

PRINT
specifies that the output medium is a printer.

SUBADDR(*data-value***)**
specifies the medium subaddress as a halfword binary value (in the range 0 through 15) which allows media of the same type, for example, "printer 1" or "printer 2", to be defined. Value 15 means a medium of any type. The default is zero.

VOLUME(*data-value***)**
specifies the name (1–6 characters) of a diskette in an outboard destination that contains the data set specified in the DESTID option.

VOLUMELENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the VOLUME option.

WPMEDIA1 through WPMEDIA4
specifies that, for each specific LUTYPE4 device, a word-processing medium is defined to relate to a specific input/output device.

Conditions

48 FUNCERR
occurs if there is an error during execution of the command. Destination selection is unaffected and other commands for the same destination may be successful.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

47 SELNERR
occurs if there is an error during destination selection. The destination is not selected and other commands for the same destination are unlikely to be successful.

Default action: terminate the task abnormally.

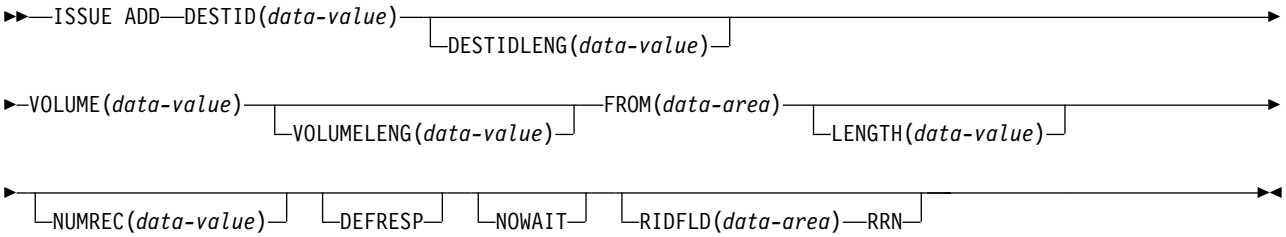
49 UNEXPIN
occurs when some unexpected or unrecognized information is received from the outboard controller.

Default action: terminate the task abnormally.

ISSUE ADD

Add a record to a data set.

ISSUE ADD



Conditions: FUNCERR, INVREQ, SELNERR, UNEXPIN

Description

ISSUE ADD adds records to a sequential or keyed direct data set in an outboard controller. The FROM option is used to specify the data to be written, and the LENGTH option specifies its length.

The RIDFLD option is only needed with this command when it applies to a DPCX/DXAM data set. In this case, it specifies the relative record number of the record to be added. When RIDFLD is used, NUMREC must be 1 (the default).

Options

DEFRESP
specifies that all terminal control commands issued as a result of the ISSUE ADD command are to request a definite response from the outboard batch program, irrespective of the specification of message integrity for the CICS task (by the system programmer).

DESTID(*data-value***)**
specifies the name (1–8 characters) of the data set in the outboard destination.

DESTIDLENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the DESTID option.

FROM(*data-area***)**
specifies the data to be written to the data set.

LENGTH(*data-value***)**
specifies the length (halfword binary value) of the data to be written. For a description of a safe upper limit, see “LENGTH options in CICS commands” on page 10.

NOWAIT
specifies that the CICS task continues processing without waiting for the ISSUE ADD command to complete. If this option is not specified, the task activity is suspended until the command is completed.

NUMREC(*data-value***)**
for a relative record data set, specifies as a halfword binary value the number of logical records to be added. Records are replaced sequentially starting with the one identified by the RIDFLD option.

For an indexed data set, NUMREC cannot be specified because only one record can be added.

RIDFLD(*data-area***)**
specifies, for a relative record data set, a 4-character field as the relative record number (starting from zero) of the record. The RRN option is also required.

For a keyed direct data set, RIDFLD should specify a key.

RRN
specifies that the record identification field specified in the RIDFLD option contains a relative record number. This option is required for a relative record data set.

VOLUME(*data-value***)**
specifies the name (1–6 characters) of a diskette in an outboard destination that contains the data set specified in the DESTID option.

VOLUMELENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the VOLUME option.

Conditions

48 FUNCERR
occurs if there is an error during execution of the command. Destination selection is unaffected and other commands for the same destination may be successful.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

47 SELNERR
occurs if there is an error during destination selection. The destination is not selected and other commands for the same destination are unlikely to be successful.

Default action: terminate the task abnormally.

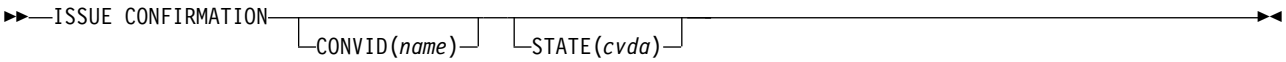
49 UNEXPIN
occurs when some unexpected or unrecognized information is received from the outboard controller.

Default action: terminate the task abnormally.

ISSUE CONFIRMATION

Issue a positive response to a SEND CONFIRM on an APPC mapped conversation.

ISSUE CONFIRMATION (APPC)



Conditions: INVREQ, NOTALLOC, SIGNAL, TERMERR

Description

ISSUE CONFIRMATION allows an application program to respond positively when the CONFIRM option has been specified on a SEND command executed by a partner transaction.

Options

CONVID(*name*)
identifies the conversation in which to send the response. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal facility (returned by a previously executed ASSIGN command).

For compatibility with earlier releases, SESSION is accepted as a synonym for CONVID. New programs should use CONVID.

If both CONVID and SESSION are omitted, the principal facility is assumed.

STATE(*cvda*)
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session on the CONVID option.

also occurs (RESP2 not set) in any of the following situations:

- ISSUE CONFIRMATION is used on a conversation that is either of the following:
  - Sync level 0
  - Not APPC mapped

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the specified CONVID value relates to a conversation that is not owned by the application.

Default action: terminate the task abnormally.

24 SIGNAL
occurs when an inbound SIGNAL data-flow control command is received from a partner transaction. EIBSIG is always set when an inbound signal is received.

Default action: ignore the condition.

81 TERMERR
occurs for a session-related error. Any action on that conversation other than a FREE causes an ATCV abend.

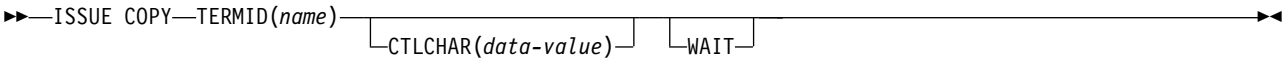
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE COPY (3270 logical)

Copy data from 3270 logical unit.

ISSUE COPY (3270 logical)



Conditions: LENGERR, NOTALLOC, TERMERR

Description

ISSUE COPY copies the format and data contained in the buffer of a specified terminal into the buffer of the terminal that started the transaction. Both terminals must be attached to the same remote control unit.

Options

CTLCHAR(*data-value*)
specifies a 1-byte copy control character (CCC) that defines the copy function. A COBOL user must specify a data area containing this character. If the option is omitted, the contents of the entire buffer (including nulls) are copied.

TERMID(*name*)
specifies the name (1–4 characters) of the terminal whose buffer is to be copied. The terminal must have been defined in the TCT.

WAIT
specifies that processing of the command must be completed before any subsequent processing is attempted.

If the WAIT option is not specified, control is returned to the application program once processing of the command has started. A subsequent input or output request (terminal control, BMS, or batch data interchange) to the terminal associated with the task causes the application program to wait until the previous request has been completed.

Conditions

22 LENGERR
occurs if an out-of-range value is supplied.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

81 TERMERR
occurs for a terminal-related error.

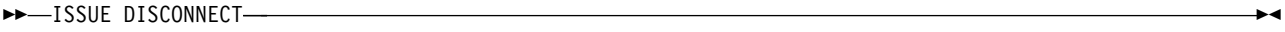
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE DISCONNECT (default)

Terminate a session between CICS and a logical unit or terminal.

ISSUE DISCONNECT (default)



Conditions: SIGNAL, TERMERR

Description

ISSUE DISCONNECT terminates sessions between CICS and the following terminals or logical units:

- 3270-display logical unit (LUTYPE2)
- 3270-printer logical unit (LUTYPE3)
- LUTYPE4 logical unit
- 3270 SCS printer logical unit
- 2260 or 2265 display station
- 3270 logical unit

- 3600 pipeline logical unit
- 3600(3601) logical unit
- 3600(3614) logical unit
- 3630 plant communication system
- 3650 interpreter logical unit
- 3650 host conversational (3270) logical unit
- 3650 host conversational (3653) logical unit
- 3650(3680) host command processor logical unit
- 3767/3770 interactive logical unit
- 3770 batch logical unit
- 3790 logical units

Conditions

For most terminal and logical unit types, ISSUE DISCONNECT raises no conditions. Exceptions are:

24 SIGNAL
occurs only for an ISSUE DISCONNECT for LUTYPE4, 3600(3601), 3767 interactive, 3770 batch, and 3790 full-function logical units.

It occurs when an inbound SIGNAL data-flow control command is received from a logical unit or session. EIBSIG is always set when an inbound signal is received.

Default action: ignore the condition.

81 TERMERR
occurs only for an ISSUE DISCONNECT for LUTYPE4 logical units.

It occurs for a terminal-related error, such as a session failure. This condition applies to SNA-connected LUs only. Because of the asynchronous nature of this condition, the application program should check, using SEND CONFIRM or SYNCPOINT, to make sure any errors still outstanding have been resolved before it relinquishes control. If you want to handle this condition, you must first issue a FREE command to free the session. If you do not do this, an INVREQ condition occurs, plus an ATCV abend if you do not handle this condition.

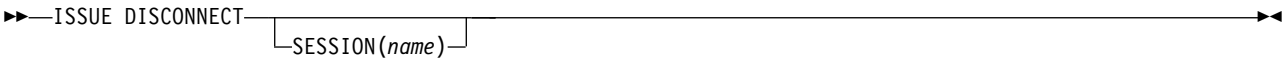
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE DISCONNECT (LUTYPE6.1)

Disconnect an LUTYPE6.1 logical unit.

ISSUE DISCONNECT (LUTYPE6.1)



Conditions: NOTALLOC, TERMERR

Description

ISSUE DISCONNECT disconnects the unit, if DISCREQ=YES is set in the TYPETERM resource definition.

Options

SESSION(*name*)
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be disconnected. If this option is omitted, the principal facility for the task is disconnected.

Conditions

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

81 TERMERR
occurs for a terminal-related error, such as a session failure.

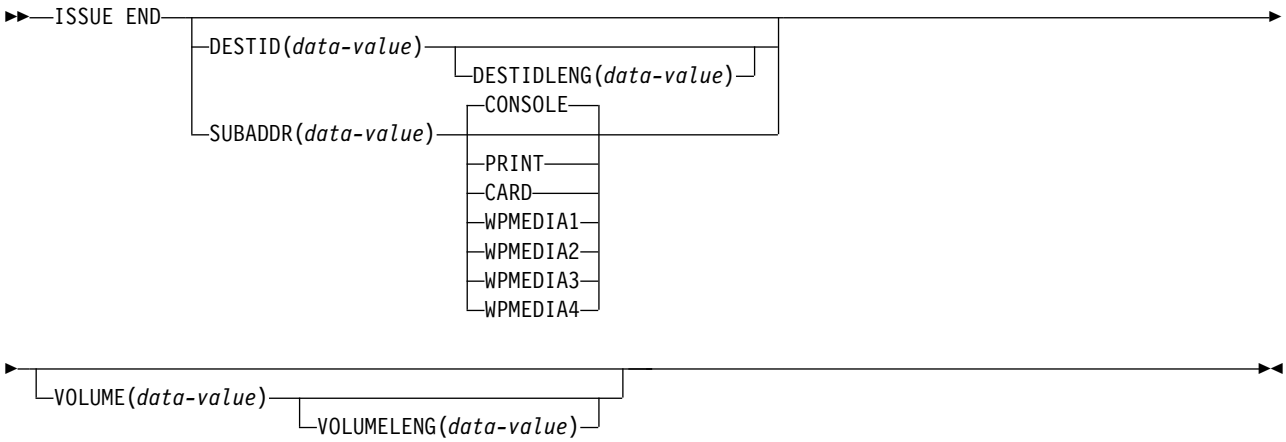
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE END

End processing of a data set.

ISSUE END



Conditions: FUNCERR, INVREQ, SELNERR, UNEXPIN

Description

ISSUE END ends communication with a data set in an outboard controller or with the selected medium. The data set specified in the DESTID option, or the selected medium, is deselected normally. The options CONSOLE, PRINT, CARD, and WPMEDIA1–4 are alternatives to DESTID and DESTIDLENG.

Options

CARD
specifies that the output medium is a card reader or card punch device. This option is not valid with DESTID and DESTIDLENG.

CONSOLE
specifies that the output medium is that provided for messages to the operator. This option is not valid with DESTID and DESTIDLENG. This refers to a programmable subsystem such as the IBM 3790 data communication system. It does not refer to a CICS or system console.

DESTID(*data-value***)**
specifies the name (1–8 characters) of the data set in the outboard destination.

DESTIDLENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the DESTID option.

PRINT
specifies that the output medium is a printer.

SUBADDR(*data-value***)**
specifies the medium subaddress as a halfword binary value (in the range

0–15) that allows media of the same type, for example, “printer 1” or “printer 2”, to be defined. Value 15 means a medium of any type. The default is zero.

VOLUME(*data-value***)**
specifies the name (1–6 characters) of a diskette in an outboard destination that contains the data set specified in the DESTID option.

VOLUMELENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the VOLUME option.

WPMEDIA1 through WPMEDIA4
specifies that, for each specific LUTYPE4 device, a word-processing medium is defined to relate to a specific input/output device.

Conditions

48 FUNCERR
occurs if there is an error during the execution of the command. Destination selection is unaffected and other commands for the same destination may be successful.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

47 SELNERR
occurs if there is an error during destination selection. The destination is not selected and other commands for the same destination are unlikely to be successful.

Default action: terminate the task abnormally.

49 UNEXPIN
occurs when some unexpected or unrecognized information is received from the outboard controller.

Default action: terminate the task abnormally.

ISSUE ENDFILE

Indicate the end-of-file condition to the 3740 data entry system.

ISSUE ENDFILE



Condition: INVREQ, NOTALLOC

Description

ISSUE ENDFILE indicates the end-of-file condition to the 3740.

Options

83 ENDOUTPUT
indicates the end-of-output condition as well as end of file.

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application attempted to send on its function shipping session, its principal facility.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

ISSUE ENDOUTPUT

Indicate the end-of-output condition to the 3740 data entry system.

ISSUE ENDOUTPUT



Condition: INVREQ, NOTALLOC

Description

ISSUE ENDOUTPUT indicates the end-of-output condition to the 3740.

Options

20 ENDFILE
indicates the end-of-file condition as well as end of output.

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application attempted to send on its function shipping session, its principal facility.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

ISSUE EODS

Send end-of-data-set function management header to the 3650 interpreter logical unit.

ISSUE EODS



Conditions: INVREQ, NOTALLOC, TERMERR

Description

ISSUE EODS issues the end-of-data-set management header.

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application attempted to send on its function shipping session, its principal facility.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

81 TERMERR
occurs for a terminal-related error, such as a session failure.

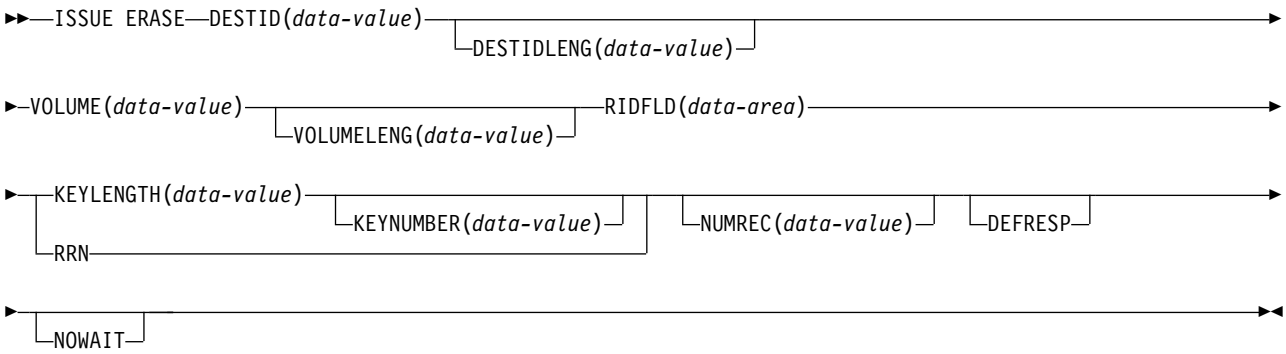
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE ERASE

Delete a record from a data set.

ISSUE ERASE



Conditions: FUNCERR, INVREQ, SELNERR, UNEXPIN

Description

ISSUE ERASE deletes a record from a keyed direct data set in an outboard controller, or erases a record from a DPCX or DXAM relative record data set.

Options

DEFRESP
specifies that all terminal control commands issued as a result of the ISSUE ERASE command are to request a definite response from the outboard batch program, irrespective of the specification of message integrity for the CICS task (by the system programmer).

DESTID(*data-value*)
specifies the name (1–8 characters) of the data set in the outboard destination.

DESTIDLENG(*data-value*)
specifies the length (halfword binary value) of the name specified in the DESTID option.

KEYLENGTH(*data-value*)
specifies the length of the key specified in the RIDFLD option, as a halfword binary value.

KEYNUMBER(*data-value*)
specifies the number, as a halfword binary value, of the index to be used to locate the record. There can be eight indexes (1–8). The default is 1. This option applies only to DPCX or DXAM and is mutually exclusive with RRN.

NOWAIT
specifies that the CICS task continues processing without waiting for the ISSUE ERASE command to complete. If this option is not specified, the task activity is suspended until the command is completed.

NUMREC(*data-value***)**
for a relative record data set, specifies as a halfword binary value the number of logical records to be deleted. Records are replaced sequentially starting with the one identified by the RIDFLD option.

For an indexed data set, NUMREC cannot be specified, because only one record is deleted.

RIDFLD(*data-area***)**
specifies the record identification field.

For a relative record data set, the RIDFLD option specifies a fullword binary integer (the relative record number of the record starting from zero); and the RRN option is used.

For an indexed data set, the RIDFLD option specifies the key that is embedded in the data. The KEYLENGTH option is also required.

RRN
specifies that the record identification field specified in the RIDFLD option contains a relative record number. If the option is not specified, RIDFLD is assumed to specify a key.

VOLUME(*data-value***)**
specifies the name (1–6 characters) of a diskette in an outboard destination that contains the data set specified in the DESTID option.

VOLUMELENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the VOLUME option.

Conditions

48 FUNCERR
occurs if there is an error during execution of the command. Destination selection is unaffected and other commands for the same destination may be successful.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

47 SELNERR
occurs if there is an error during destination selection. The destination is not selected and other commands for the same destination are unlikely to be successful.

Default action: terminate the task abnormally.

49 UNEXPIN
occurs when some unexpected or unrecognized information is received from the outboard controller.

Default action: terminate the task abnormally.

ISSUE ERASEAUP

Erase all unprotected fields of a 3270 buffer.

ISSUE ERASEAUP



Conditions: INVREQ, NOTALLOC, TERMERR

Description

ISSUE ERASEAUP erases unprotected fields by:

1. Clearing all unprotected fields to nulls (X'00')
2. Resetting modified data tags in each unprotected field to zero
3. Positioning the cursor to the first unprotected field
4. Restoring the keyboard

You can use the ISSUE ERASEAUP command for the following types of 3270 logical units:

- 3270-display logical unit (LUTYPE2)
- 3270-printer logical unit (LUTYPE3)
- 3270 logical unit
- 3650 host conversational (3270) logical unit
- 3790 (3270-display) logical unit
- 3790 (3270-printer) logical unit

Options

WAIT
ensures that the erase is completed before control returns to the application program. If you omit WAIT, control returns to the application program as soon as ISSUE ERASEAUP starts processing.

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

81 TERMERR
occurs if there is a terminal-related error.

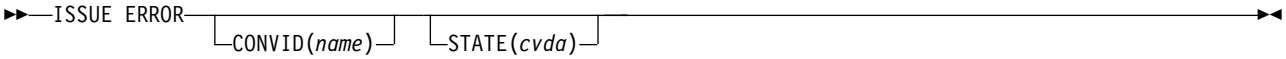
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE ERROR

Inform APPC mapped conversation partner of error.

ISSUE ERROR (APPC)



Conditions: INVREQ, NOTALLOC, SIGNAL, TERMERR

Description

ISSUE ERROR allows an application program to inform a process in a connected APPC system that some program-detected error has occurred. For example, a remote CICS application is notified by having EIBERR set, with EIBERRCD=X'0889'. The actions required to recover from the error are the responsibility of logic contained in both application programs. The application program can use this command to respond negatively when the CONFIRM option has been specified on a SEND command executed by a process in a connected APPC system.

Options

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal facility (returned by a previously executed ASSIGN command).

For compatibility with earlier releases, SESSION is accepted as a synonym for CONVID. New programs should use CONVID.

If both CONVID and SESSION are omitted, the principal facility is assumed.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE

- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session on the CONVID option.

also occurs (RESP2 not set) in any of the following situations:

- The command is not valid for the APPC conversation type in use.
- The command is issued against a CPI-Communications conversation.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the specified CONVID value does not relate to a conversation owned by the application.

Default action: terminate the task abnormally.

24 SIGNAL
occurs when an inbound SIGNAL data-flow control command is received from a partner transaction. EIBSIG is always set when an inbound signal is received.

Default action: ignore the condition.

81 TERMERR
occurs for a session-related error. Any action on that conversation other than a FREE command causes an ATCV abend.

A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE LOAD

Specify the name of a program on 3650 interpreter logical unit.

ISSUE LOAD



Conditions: NONVAL, NOTALLOC, NOSTART, TERMERR

Description

ISSUE LOAD specifies the name of the 3650 application program that is to be loaded.

Options

CONVERSE
specifies that the 3650 application program is able to communicate with the host processor. If this option is not specified, the 3650 application program cannot communicate with the host processor.

PROGRAM(*name***)**
specifies the name (1–8 characters) of the 3650 application program that is to be loaded.

Conditions

09 NONVAL
occurs if the 3650 application program name is not valid.

Default action: terminate the task abnormally.

10 NOSTART
occurs if the 3651 is unable to initiate the requested 3650 application program.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

81 TERMERR
occurs for a terminal-related error.

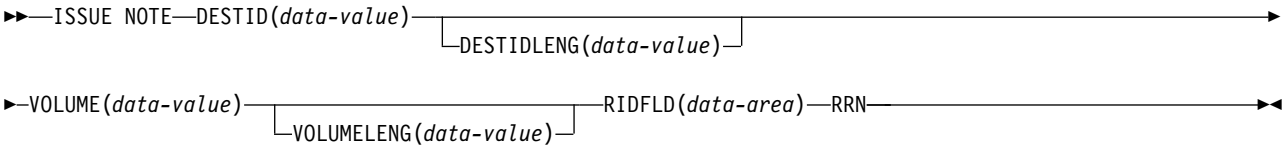
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE NOTE

Request next record number.

ISSUE NOTE



Conditions: FUNCERR, INVREQ, SELNERR, UNEXPIN

Description

ISSUE NOTE requests the number of the next record. It finds the relative record number of the next record in an addressed direct data set. The number is returned in the data area specified in the RIDFLD option. The RRN option must be specified, because a relative record number is involved.

Options

DESTID(*data-value***)**
specifies the name (1–8 characters) of the data set in the outboard destination.

DESTIDLENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the DESTID option.

RIDFLD(*data-area***)**
specifies as a 4-character field a data area the relative record number of the next record is returned in.

RRN
specifies that the record identification field specified in the RIDFLD option contains a relative record number.

VOLUME(*data-value***)**
specifies the name (1–6 characters) of a diskette in an outboard destination that contains the data set specified in the DESTID option.

VOLUMELENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the VOLUME option.

Conditions

48 FUNCERR
occurs if there is an error during execution of the command. Destination selection is unaffected and other commands for the same destination may be successful.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

47 SELNERR
occurs if there is an error during destination selection. The destination is not selected and other commands for the same destination are unlikely to be successful.

Default action: terminate the task abnormally.

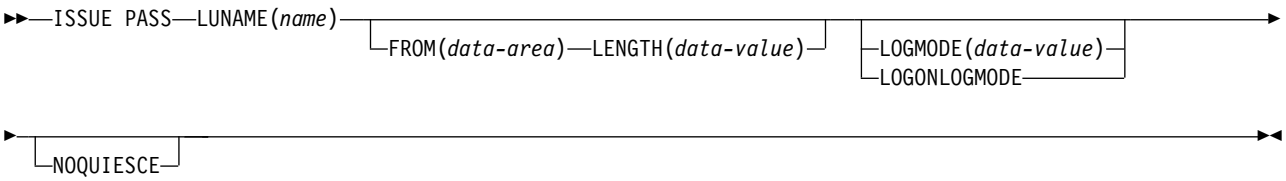
49 UNEXPIN
occurs when some unexpected or unrecognized information is received from the outboard controller.

Default action: terminate the task abnormally.

ISSUE PASS

z/OS Communications Server application routing.

ISSUE PASS



Conditions: INVREQ, LENGERR, NOTALLOC

Description

ISSUE PASS disconnects the terminal from CICS after the task has terminated, and transfers it to the z/OS Communications Server application defined in the LUNAME option.

This command requires that AUTH=PASS is coded on the z/OS Communications Server APPL macro for the CICS terminal-owning system that issues it, with DISCREQ=YES or RELREQ=YES in the RDO TYPETERM resource definition for any terminal where this function might be used.

If the LUNAME specified is the name of another CICS system, you can use the EXTRACT LOGONMSG command to access the data referred to by this command.

Because of a z/OS Communications Server limitation, the maximum length of the user data is restricted to 255 bytes.

Note: The system initialization parameter CLSDSTP=NOTIFY | NONOTIFY allows you to have the node error program (NEP) and the console notified of whether the PASS was successful or not. The NEP can be coded to reestablish a session ended by an unsuccessful PASS. For programming information about how to do this, see the section about NEP in Writing a node error program.

Options

FROM(*data-area***)**
specifies the data area containing the logon user data that is to be passed to the application named in the LUNAME option. This option may be omitted if ATTACHID is specified on an LUTYPE6.1 command.

LENGTH(*data-value***)**
specifies the length, as a halfword binary value, of the data issued.

LOGMODE(*data-value***)**
specifies the name (1–8 characters) of the z/OS Communications Server logon mode table entry used by z/OS Communications Server to establish the new session.

LOGONLOGMODE
specifies that the new session is to be established with the z/OS Communications Server logon mode table entry in use when the session logged on.

Note: The logmode name saved is taken from the X'0D' control vector in the z/OS Communications Server CINIT. This is the logmode name known in this system.

If persistent sessions (PSDINT=nnn in the SIT) is in use, then the TYPETERM definition for any terminal to be ISSUE PASSed should use RECOVOPTION(NONE), because the logon LOGMODE name is not recovered across a persistent sessions restart.

If neither LOGMODE nor LOGONLOGMODE is supplied, the new session will be established with the default LOGMODE.

LUNAME(*name*)
specifies the name (1–8 characters) of the z/OS Communications Server application to which the terminal is to be passed.

NOQUIESCE
specifies that the user can choose to recover from certain pass failures.

Conditions

16 INVREQ
occurs if the command is not valid for the logical unit in use.

Default action: terminate the task abnormally.

22 LENGERR
occurs if an out-of-range value is supplied in the LENGTH option.

Default action: terminate the task abnormally.

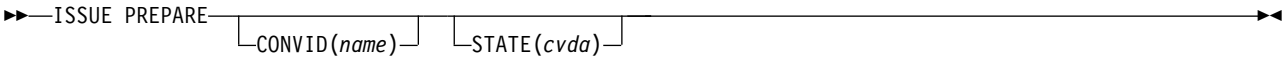
61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

ISSUE PREPARE

Issue the first flow of a syncpoint request on an APPC mapped conversation.

ISSUE PREPARE (APPC)



Conditions: INVREQ, NOTALLOC, TERMERR

Description

ISSUE PREPARE applies only to distributed transaction processing over APPC links. It enables a syncpoint initiator to prepare a syncpoint slave for syncpointing

by sending only the first flow (prepare-to-commit) of the syncpoint exchange. Depending on the reply from the syncpoint slave, the initiator can proceed with the syncpoint by issuing a SYNCPOINT command, or initiate back-out by issuing a SYNCPOINT ROLLBACK command.

Options

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal facility (returned by a previously executed ASSIGN command).

For compatibility with earlier releases, SESSION is accepted as a synonym for CONVID. New programs should use CONVID.

If both CONVID and SESSION are omitted, the principal facility is assumed.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

also occurs (RESP2 not set) in any of the following situations:

- The conversation is not an APPC mapped conversation.
- The conversation state is not valid for the request.
- The sync level of the conversation is not 2.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the CONVID value in the command does not relate to a conversation that is owned by the application.

Default action: terminate the task abnormally.

81 TERMERR
occurs for a session-related error. Any action on that conversation other than a FREE causes an ATCV abend.

A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE PRINT

Print displayed data on first available printer.

ISSUE PRINT



Conditions: INVREQ, NOTALLOC, TERMERR

Description

ISSUE PRINT prints displayed data on the first available printer that can respond to a print request.

ISSUE PRINT can be used on a number of logical units, using the printers defined below:

- For a 3270 logical unit or a 3650 host conversational (3270) logical unit, the printer must be defined by the PRINTER or ALTPRINTER options on the RDO TERMINAL resource definition, or by a printer supplied by the autoinstall user program.
- For a 3270-display logical unit with the PTRADAPT feature, used with a 3274 or 3276, the printer is allocated by the printer authorization matrix. The PTRADAPT feature is enabled by specifying DEVICE=LUTYPE2 and PRINTADAPTER=YES on the RDO TYPETERM resource definition.
- For a 3790 (3270-display) logical unit, the printer is allocated by the 3790.

  The printer must be in service, not currently attached to a task, and owned by the same CICS system that owns the terminal running the transaction.

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

81 TERMERR
occurs for a terminal-related error.

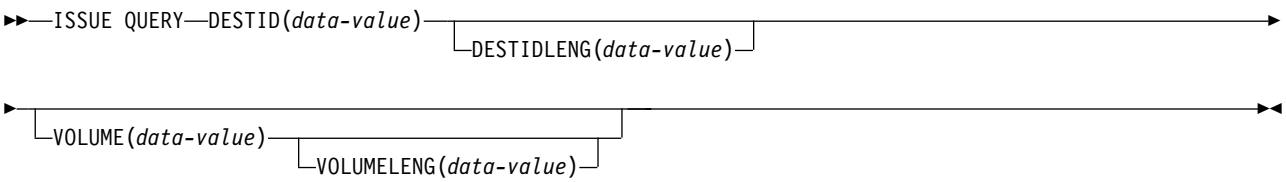
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE QUERY

Interrogate a data set.

ISSUE QUERY



Conditions: FUNCERR, INVREQ, SELNERR, UNEXPIN

Description

ISSUE QUERY interrogates a data set. It is used to request that a sequential data set in an outboard controller be transmitted to the host system. The application program should either follow this command with ISSUE RECEIVE commands to get the resulting inbound data, or terminate the transaction to allow CICS to start a new transaction to process the data.

Options

DESTID(*data-value***)**
specifies the name (1–8 characters) of the data set in the outboard destination.

DESTIDLENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the DESTID option.

VOLUME(*data-value***)**
specifies the name (1–6 characters) of a diskette in an outboard destination that contains the data set specified in the DESTID option.

VOLUMELENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the VOLUME option.

Conditions

48 FUNCERR
occurs if there is an error during execution of the command. Destination selection is unaffected and other commands for the same destination may be successful.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

47 SELNERR
occurs if there is an error during destination selection. The destination is not selected and other commands for the same destination are unlikely to be successful.

Default action: terminate the task abnormally.

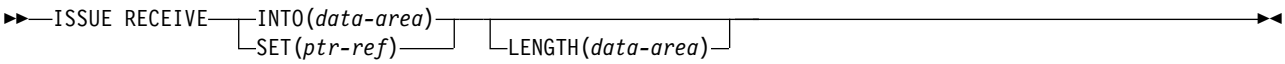
49 UNEXPIN
occurs when some unexpected or unrecognized information is received from the outboard controller.

Default action: terminate the task abnormally.

ISSUE RECEIVE

Read a record from a data set.

ISSUE RECEIVE



Conditions: DSSTAT, EOC, EODS, INVREQ, LENGERR, UNEXPIN

Description

ISSUE RECEIVE reads a sequential data set in an outboard controller.

The INTO option specifies the area into which the data is to be placed. The LENGTH option must specify a data area that contains the maximum length of record that the program accepts. If the record length exceeds the specified maximum length, the record is truncated and the LENGERR condition occurs. After the retrieval operation, the data area specified in the LENGTH option is set to the record length (before any truncation occurred).

Alternatively, a pointer reference can be specified in the SET option. CICS then acquires an area of sufficient size to hold the record, and sets the pointer reference to the address of that area. After the retrieval operation, the data area specified in the LENGTH option is set to the record length.

The outboard controller might not send the data from the data set specified in the ISSUE QUERY command. The ASSIGN command must be used to get the value of DESTID (which identifies the data set that has been transmitted) and the value of DESTIDLENG (which is the length of the identifier in DESTID).

Options

INTO(*data-area***)**
specifies the receiving field for the data read from the data set.

If you specify the ISSUE RECEIVE command with the INTO option, the parameter must be a data area that specifies the maximum length of data that the program is prepared to handle. If the value specified is less than zero, zero is assumed. If the length of the data exceeds the value specified, the data is truncated to that value and the LENGERR condition occurs. On completion of the retrieval operation, the data area is set to the original length of the data.

LENGTH(*data-area***)**
specifies the length (halfword binary value) of the data received.

If you have specified SET, you must also specify LENGTH.

SET(*ptr-ref***)**
specifies the pointer reference that is to be set to the address location of the data read from the data set.

If you specify the SET option, the parameter must be a data area. On completion of the retrieval operation, the data area is set to the length of the data.

If DATALOCATION(ANY) is associated with the application program, the address of the data can be above or below the 16MB line.

If DATALOCATION(BELOW) is associated with the application program, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

If you have specified SET, you must also specify LENGTH.

Conditions

46 DSSTAT
occurs when the destination status changes in one of the following ways:

- The data stream is abended.
- The data stream is suspended.

Default action: terminate the task abnormally.

06 EOC
occurs if the request/response unit (RU) is received with the end-of-chain (EOC) indicator set. Field EIBEOC also contains this indicator.

Default action: ignore the condition.

05 EODS
occurs when the end of the data set is encountered.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

22 LENGERR
occurs if the length of the retrieved data is greater than the value specified by the LENGTH option.

Default action: terminate the task abnormally.

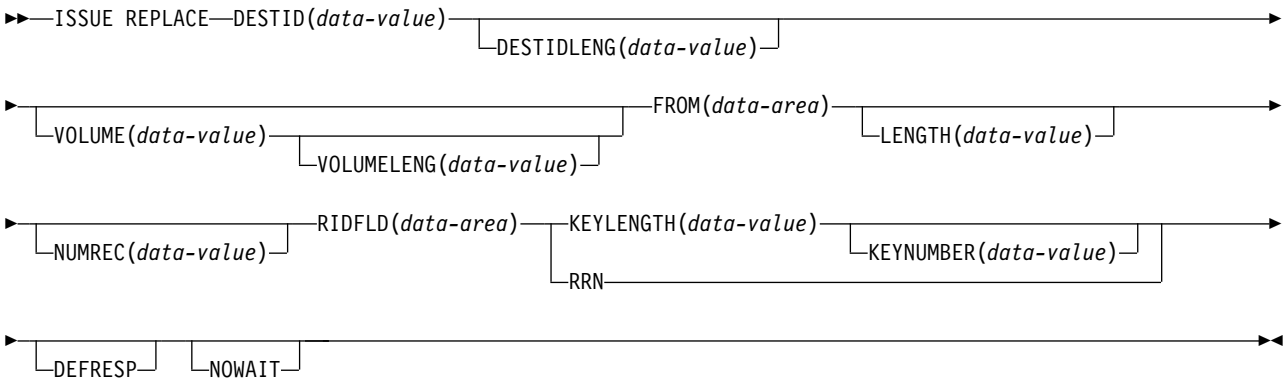
49 UNEXPIN
occurs when some unexpected or unrecognized information is received from the outboard controller.

Default action: terminate the task abnormally.

ISSUE REPLACE

Update a record in a data set.

ISSUE REPLACE



Conditions: FUNCERR, INVREQ, SELNERR, UNEXPIN

Description

ISSUE REPLACE updates (replaces) a record in either a relative (addressed direct) or an indexed (keyed direct) data set in an outboard controller.

Options

DEFRESP
specifies that all terminal control commands issued as a result of the ISSUE REPLACE command request a definite response from the outboard batch program, irrespective of the specification of message integrity for the CICS task (by the system programmer).

DESTID(*data-value***)**
specifies the name (1–8 characters) of the data set in the outboard destination.

DESTIDLENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the DESTID option.

FROM(*data-area***)**
specifies the data that is to be written to the data set.

KEYLENGTH(*data-value***)**
specifies the length (halfword binary value) of the key specified in the RIDFLD option.

KEYNUMBER(*data-value***)**
specifies the number, as a halfword binary value, of the index to be used to locate the record. There can be eight indexes (1 through 8). The default is 1. This option applies only to DPCX/DXAM and is mutually exclusive with RRN.

LENGTH(*data-value***)**
specifies the length (halfword binary value) of the data to be written.

NOWAIT
specifies that the CICS task continues processing without waiting for the ISSUE REPLACE command to complete. If this option is not specified, the task activity is suspended until the command is completed.

NUMREC(*data-value***)**
for a relative data set, specifies as a halfword binary value the number of logical records to be replaced. Records are replaced sequentially starting with the one identified by the RIDFLD option.

For an indexed data set, NUMREC cannot be specified because only one record is replaced.

RIDFLD(*data-area***)**
specifies the record identification field.

For a relative record data set, the RIDFLD option specifies a fullword binary integer (the relative record number of the record starting from zero); and the RRN option is used.

For an indexed data set, the RIDFLD option specifies the key that is embedded in the data specified by the FROM option. The KEYLENGTH option is also required.

RRN
specifies that the record identification field specified in the RIDFLD option contains a relative record number. This option is required for a relative record data set.

If the option is not specified, RIDFLD is assumed to specify a key.

VOLUME(*data-value***)**
specifies the name (1–6 characters) of a diskette in an outboard destination that contains the data set specified in the DESTID option.

VOLUMELENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the VOLUME option.

Conditions

48 FUNCERR
occurs if there is an error during execution of the command. Destination selection is unaffected and other commands for the same destination may be successful.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

47 SELNERR
occurs if there is an error during destination selection. The destination is not selected and other commands for the same destination are unlikely to be successful.

Default action: terminate the task abnormally.

49 UNEXPIN
occurs when some unexpected or unrecognized information is received from the outboard controller.

Default action: terminate the task abnormally.

ISSUE RESET

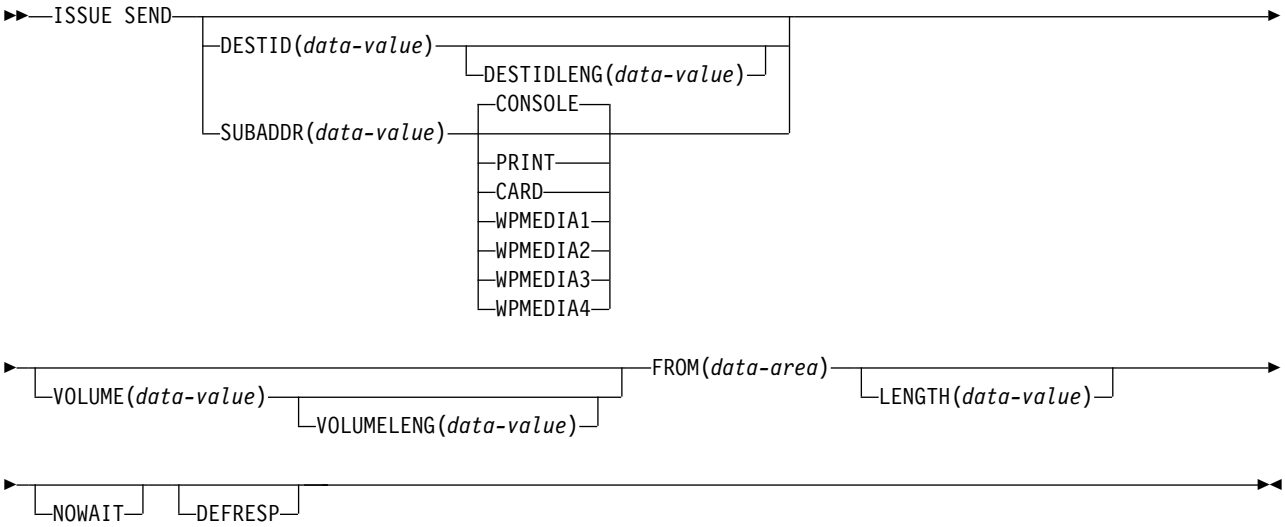
Relinquish use of a telecommunication line.

This command is supported for compatibility with earlier releases of CICS. It is superseded by the ISSUE DISCONNECT command, which you are recommended to use instead.

ISSUE SEND

Send data to a named data set or to a selected medium.

ISSUE SEND



Conditions: FUNCERR, IGREQCD, INVREQ, SELNERR, UNEXPIN

Description

ISSUE SEND sends data to a named data set in an outboard controller, or to a selected medium in a batch logical unit or an LUTYPE4 logical unit. The options CONSOLE, PRINT, CARD, and WPMEDIA1–4 are alternatives to DESTID and DESTIDLENG.

Options

CARD
specifies that the output medium is a card reader or card punch device. This option is not valid with DESTID and DESTIDLENG.

CONSOLE
specifies that the output medium is that provided for messages to the operator. This option is not valid with DESTID and DESTIDLENG. This refers to a programmable subsystem such as the IBM 3790 data communication system. It does not refer to a CICS or system console.

DEFRESP
specifies that all terminal control commands issued as a result of the ISSUE SEND command request a definite response from the outboard batch program, irrespective of the specification of message integrity for the CICS task (by the system programmer).

DESTID(*data-value***)**
specifies the name (1–8 characters) of the data set in the outboard destination.

DESTIDLENG(*data-value***)**
specifies the length (halfword binary value) of the name specified in the DESTID option.

FROM(*data-area***)**
specifies the data to be written to the data set.

LENGTH(*data-value***)**
specifies a halfword binary value that is the length of the data to be written.

NOWAIT
specifies that the CICS task continues processing without waiting for the ISSUE SEND command to complete. If this option is not specified, the task activity is suspended until the command is completed.

PRINT
specifies that the output is to the print medium.

SUBADDR(*data-value***)**
specifies the medium subaddress as a halfword binary value (in the range 0–15) that allows media of the same type, for example, “printer 1” or “printer 2”, to be defined. Value 15 means a medium of any type. The default is zero.

VOLUME(*data-value***)**
specifies the name (1–6 characters) of a diskette in an outboard destination that contains the data set specified in the DESTID option.

VOLUMELENG(*data-value***)**
specifies the length of the name specified in the VOLUME option as a halfword binary value.

WPMEDIA1 through WPMEDIA4
specifies that for each specific LUTYPE4 device, a word processing medium is defined to relate to a specific input/output device.

Conditions

48 FUNCERR
occurs if there is an error during execution of the command. Destination selection is unaffected and other commands for the same destination may be successful.

Default action: terminate the task abnormally.

57 IGREQCD
occurs when an attempt is made to execute an ISSUE SEND command after a SIGNAL RCD data-flow control code has been received from an LUTYPE4 logical unit.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

47 SELNERR
occurs if there is an error during destination selection. The destination is not selected and other commands for the same destination are unlikely to be successful.

Default action: terminate the task abnormally.

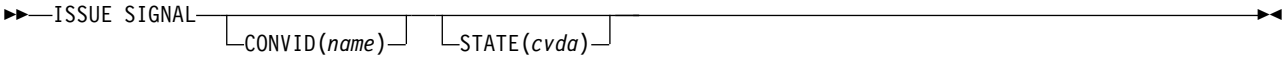
49 UNEXPIN
occurs when some unexpected or unrecognized information is received from the outboard controller.

Default action: terminate the task abnormally.

ISSUE SIGNAL (APPC)

Request change of direction from sending transaction on an APPC mapped conversation.

ISSUE SIGNAL (APPC)



Conditions: INVREQ, NOTALLOC, TERMERR

Description

ISSUE SIGNAL, in a transaction in receive mode, signals to the sending transaction that a mode change is needed. It raises the SIGNAL condition on the next SEND, RECEIVE, or CONVERSE command executed in the sending transaction, and a previously executed HANDLE CONDITION command for this condition can be used either to take some action, or to ignore the request.

Options

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal facility (returned by a previously executed ASSIGN command).

For compatibility with earlier releases, SESSION is accepted as a synonym for CONVID. New programs should use CONVID.

If both CONVID and SESSION are omitted, the principal facility is assumed.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

also occurs (RESP2 not set) in any of the following situations:

- The command has been used on an APPC conversation that is not using the EXEC CICS interface, or is not a mapped conversation.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the specified CONVID value does not relate to a conversation that is owned by the application.

Default action: terminate the task abnormally.

81 TERMERR
occurs for a session-related error. Any action on that conversation other than a FREE causes an ATCV abend.

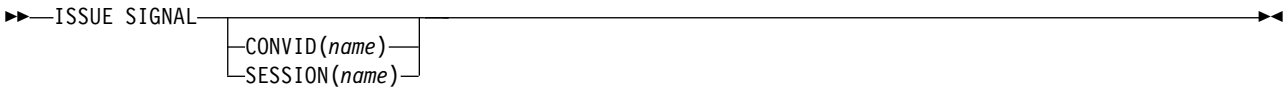
A CANCEL TASK request by a user node error program (NEP) may cause TERMERR if the task has an outstanding terminal control request when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE SIGNAL (LUTYPE6.1)

Request change of direction from sending transaction on LUTYPE6.1 conversation.

ISSUE SIGNAL (LUTYPE6.1)



Conditions: NOTALLOC, TERMERR

Description

ISSUE SIGNAL, in a transaction in receive mode, signals to the sending transaction that a mode change is needed. It raises the SIGNAL condition on the next SEND, RECEIVE, or CONVERSE command executed in the sending transaction, and a previously executed HANDLE CONDITION command for this condition can be used either to take some action, or to ignore the request.

If both CONVID and SESSION are omitted, the principal facility for the task is used.

Options

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal facility (returned by a previously executed ASSIGN command).

SESSION(*name***)**
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used.

Conditions

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

81 TERMERR
occurs for a session-related error. Any action on that conversation other than a FREE causes an ATCV abend.

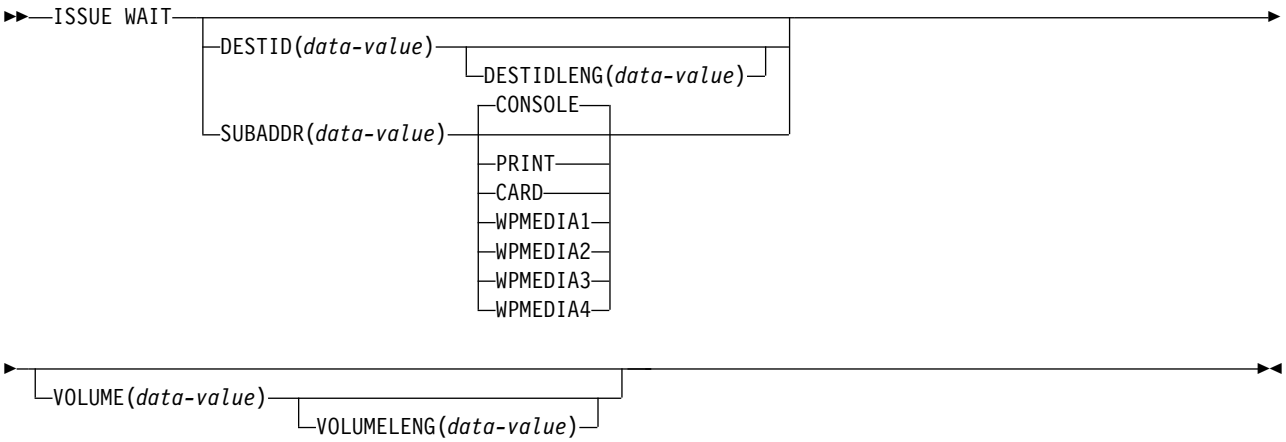
A CANCEL TASK request by a user node error program (NEP) may cause TERMERR if the task has an outstanding terminal control request when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

ISSUE WAIT

Wait for an operation to be completed.

ISSUE WAIT



Conditions: FUNCERR, INVREQ, SELNERR, UNEXPIN

Description

ISSUE WAIT suspends task activity until the previous batch data interchange command is completed. This command is meaningful only when it follows an ISSUE ADD, ISSUE ERASE, ISSUE REPLACE, or ISSUE SEND command. The options CONSOLE, PRINT, CARD, and WPMEDIA1–4 are alternatives to DESTID and DESTIDLENG.

Options

CARD
specifies that the output medium is a card reader or card punch device. This option is not valid with DESTID and DESTIDLENG.

CONSOLE
specifies that the output medium is that provided for messages to the operator. This option is not valid with DESTID and DESTIDLENG.

This refers to a programmable subsystem such as the IBM 3790 data communication system. It does not refer to a CICS or system console.

DESTID(*data-value*)
specifies the name (1–8 characters) of the data set in the outboard destination.

DESTIDLENG(*data-value*)
specifies the length (halfword binary value) of the name specified in the DESTID option.

PRINT
specifies that the output is to the print medium.

SUBADDR(*data-value*)
specifies the medium subaddress as a halfword binary value (in the range

0–through 15) that allows media of the same type, for example, “printer 1” or “printer 2”, to be defined. Value 15 means a medium of any type. The default is zero.

VOLUME(*data-value***)**
specifies the name (1–6 characters) of a diskette in an outboard destination that contains the data set specified in the DESTID option.

VOLUMELENG(*data-value***)**
specifies the length of the name specified in the VOLUME option as a halfword binary value.

WPMEDIA1 through WPMEDIA4
specifies that, for each specific LUTYPE4 device, a word-processing medium is defined to relate to a specific input/output device.

Conditions

48 FUNCERR
occurs if there is an error during execution of the command. Destination selection is unaffected and other commands for the same destination may be successful.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

47 SELNERR
occurs if there is an error during destination selection. The destination is not selected and other commands for the same destination are unlikely to be successful.

Default action: terminate the task abnormally.

49 UNEXPIN
occurs when some unexpected or unrecognized information is received from the outboard controller.

Default action: terminate the task abnormally.

JOURNAL

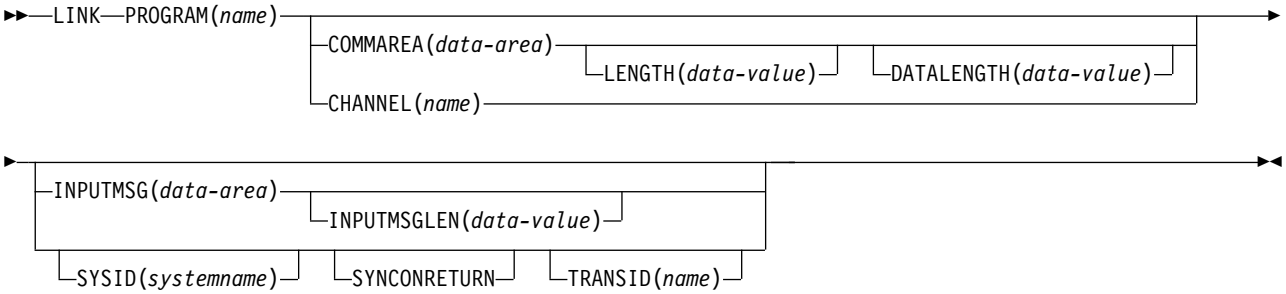
Create a journal record.

This command is supported for compatibility with earlier releases of CICS. It is superseded by the WRITE JOURNALNAME command, which you are recommended to use instead.

LINK

Link to another program expecting return.

LINK



Conditions: CHANNELERR, INVREQ, LENGERR, NOTAUTH, PGMIDERR, RESUNAVAIL, ROLLEDBACK, SYSIDERR, TERMERR

This command is threadsafe when it is used to link to a program in a local CICS region, or in a remote CICS region over an IPIC connection. It is non-threadsafe when it is used to link to a program in a remote CICS region over another type of connection.

Description

LINK passes control from an application program at one logical level to an application program at the next lower logical level.

If the requested program is not defined to CICS, and AUTOINSTALL is active, CICS supplies a definition for the program. If this definition is local, and the linked-to program is not already in main storage, CICS loads it.

In some circumstances, the linked-to program might reside on another CICS region; see “Distributed program link” on page 325.

This command operates in the current application context. If the command is issued by a program that is running under a task for an application deployed on a platform, CICS searches first for the named program in the private program directory for the application. If the named program is not found there, CICS then searches the public program directory.

When this command is used to link to a program that is declared as an application entry point for an application deployed on a platform, the CICS bundle where the application entry point is declared must have a status of AVAILABLE. The link is made to the highest numbered version of the application that is installed, enabled, and available. To link to a specified version of an application deployed on a platform, use the INVOKE APPLICATION command instead of the LINK command.

When the RETURN command runs in the linked-to program, control is returned to the program initiating the link at the next sequential executable instruction.

The external CICS interface (EXCI) provides a LINK command that performs all six commands of the interface in one invocation. See The EXCI CALL interface for information about EXCI.

The linked-to program operates independently of the program that issues the LINK command with regard to handling conditions, attention identifiers, abends, and execution key. For example, the effects of **HANDLE CONDITION** commands in the

linking program are not inherited by the linked-to program, but the original **HANDLE CONDITION** commands are restored on return to the linking program. See Using the HANDLE CONDITION command for more information and an illustration of the concept of logical levels.

You can use the **HANDLE ABEND** command to deal with abnormal terminations in other link levels. See Using the HANDLE CONDITION command for further details about the relationship between **LINK** and **HANDLE ABEND**.

Distributed program link

In any of the following cases, the link is a *distributed program link* (DPL):

- You specify a remote region name on the SYSID option, with or without the associated TRANSID and SYNCONRETURN options.
- The REMOTESYSTEM option on the installed PROGRAM definition specifies the name of a remote region.
- The installed program definition specifies DYNAMIC(YES), or there is no installed program definition, and the dynamic routing program routes the link request to a remote region.

In response to a distributed program link, the local CICS region (the *client region*) ships the link request to the remote region (the *server region*). The server region runs the linked-to program (the server program) on behalf of the program that issues the link request (the client program).

The SYSID and INPUTMSG options are mutually exclusive. If you specify both options on a LINK command, the translator issues error message DFH7230E indicating conflicting options.

A server program running in the server region is restricted to a DPL subset of the CICS API. In summary, the server program cannot issue these commands:

- Terminal control commands that reference the principal facility
- Options of ASSIGN that return terminal attributes
- BMS commands
- Sign-on and sign-off commands
- Batch data interchange commands
- Commands that address the TCTUA

For details of the restricted DPL subset of the API, see Exception conditions for LINK command in Reference -> Application development.

If a server program abends, the abend code is returned to the client program. If the client program is not written to handle the abend returned by the server program, the client program abends with the same abend code returned by the server program.

You cannot use DPL to link to the CICS master terminal program, DFHEMTA, or to the RDO program, DFHEDAP. The addresses passed as parameters to DFHEMTA and DFHEDAP are valid only in the region that issues the EXEC CICS LINK command, which means that you cannot route a DFHEMTA or DFHEDAP request to a remote CICS.

Important: For examples of the use of the LINK command when the linked program is remote, see Distributed Program Link (DPL). For information about writing a dynamic routing program, see Routing DPL requests dynamically.

Options

CHANNEL(*name***)**
Specifies the name (1 - 16 characters) of a channel that is to be made available to the called program. The acceptable characters are A - Z a - z 0 - 9 $ @ # / % & ? ! : | " = ¬ , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters. If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

Channel names are always in EBCDIC. The set of allowed characters for channel names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be shipped between regions, it is advisable to restrict the characters used to name them to A-Z a-z 0-9 & : = , ; < > . - and _.

You can specify the channel name `DFHTRANSACTION` to use a transaction channel. A transaction channel does not go out of scope when the link level changes: it is always accessible in the transaction. For more information, see Channels and containers.

The program that issues the LINK command can do one of the following:
- Have already created the channel by using one or more **PUT CONTAINER CHANNEL** or **PUT64 CONTAINER** commands.
- Specify its current channel, by name.
- Name a channel that does not currently exist. A new empty channel is created.

COMMAREA(*data-area***)**
Specifies a communication area that is to be made available to the called program. In this option the data area is passed, and you must give it the name DFHCOMMAREA in the receiving program. COMMAREA is not supported for LINK to Liberty. See Passing data to other programs .

DATALENGTH(*data-value***)**
Specifies a halfword binary value that is the length of a contiguous area of storage, from the start of the COMMAREA, to be passed to the called program. For a remote LINK request, if the amount of data being passed in a COMMAREA is small, but the COMMAREA itself is large so that the linked-to program can return the requested data, specify DATALENGTH in the interest of performance.

The value of DATALENGTH is checked only when the LINK request is remote or dynamic. It is not checked for static local links.

DATALENGTH cannot be used at the same time as INPUTMSG.

INPUTMSG(*data-area***)**
Specifies data to be supplied to the called program when it first issues a RECEIVE command. This data remains available until the execution of a RECEIVE or RETURN command. A called program can call a further program and so on, creating a chain of linked programs. If a linked-to chain exists, CICS supplies the INPUTMSG data to the first RECEIVE command run in the chain. If control returns to the program that issued the LINK with INPUTMSG before the INPUTMSG data has been accepted by a RECEIVE command, CICS

assumes that a RECEIVE command has been issued. In this situation, the original INPUTMSG data is no longer available.

INPUTMSG cannot be used at the same time as DATALENGTH. INPUTMSG is not supported for LINK to Liberty.

For more information about the INPUTMSG option, see INPUTMSG .

INPUTMSGLEN(*data-value***)**
Specifies a halfword binary value to be used with INPUTMSG.

LENGTH(*data-value***)**
Specifies a halfword binary value that is the length in bytes of the COMMAREA (communication area). This value must not exceed 24 KB if the COMMAREA is to be passed between any two CICS servers (for any combination of product, version, and release). This limit allows for the COMMAREA and space for headers.

Ensure that the value you specify matches the length of the data being passed in the COMMAREA. Do not specify 0 (zero) for LENGTH, because the resulting behavior is unpredictable and the EXEC CICS LINK command might fail.

When you use a COMMAREA to pass data, the program that is linked to must verify that the EIBCALEN field in the EIB of the task matches what the program expects. Discrepancies might result in storage violations or system failures. For more information, see COMMAREA.

PROGRAM(*name***)**
Specifies the identifier (1 - 8 characters) of the program to which control is to be passed unconditionally.

In any of the following cases, the linked-to program is a server program in a remote region:

- The SYSID option specifies a remote region.
- The REMOTESYSTEM option on the installed PROGRAM definition specifies the name of a remote region.
- The installed program definition specifies DYNAMIC(YES), or there is no installed program definition, and the dynamic routing program routes the link request to a remote region.

Note the use of quotes:
`PROGX` is in quotes because it is the program name.

```
EXEC CICS LINK PROGRAM('PROGX')
```

`DAREA` is not in quotes because it is the name of a data area that contains the

```
EXEC CICS LINK PROGRAM(DAREA)
```

actual program name. It a data area is used to contain the program name, this data area must be defined as an 8 byte field in working storage.

Note: When a link is made to a CICS 3270 program that is to run under the Link3270 bridge mechanism, the PROGRAM name must be DFHL3270, not the name of the target 3270 program.

SYNCONRETURN
Specifies that the server region named on the SYSID option is to take a sync point on successful completion of the server program.

Changes to recoverable resources made by the server program are committed or rolled back independently of changes to recoverable resources made by the client program issuing the LINK request or changes made by the server in any subsequent LINK.

- The NORMAL condition is returned if changes to recoverable resources are committed before return from the server program.
- The ROLLEDBACK condition is returned if changes to recoverable resources are rolled back before return from the server program.
- The TERMERR condition is raised following failure of the communications link or the system in which the server program is running. The client program handles the condition and ensures that data consistency is restored.

SYNCONRETURN is only applicable to remote links, it is ignored if the link is local.

SYSID(*systemname***)**
Specifies the system name of a CICS server region to where the program link request is to be routed.

If you do not specify a remote system name in the SYSID option or omit the SYSID option, CICS uses the REMOTESYSTEM attribute defined in the installed PROGRAM definition. If you specify a local system name in the SYSID option or the REMOTESYSTEM attribute, CICS ignores the name.

A remote system name specified on the SYSID option takes priority over any remote system name specified on the PROGRAM resource definition or returned by the dynamic routing program.

TRANSID(*name***)**
Specifies the name of the mirror transaction that the remote region is to attach and under which it is to run the server program.

The transaction name that you specify on the LINK command takes priority over any transaction specified on the program resource definition. While you can specify your own name for the mirror transaction initiated by DPL requests, the transaction must be defined in the server region, and the transaction definition must specify the mirror program, DFHMIRS.

If you omit the TRANSID option, reference is made to PROGRAM resource definitions held locally if the installed PROGRAM definition specifies remote attribute DYNAMIC(YES). Otherwise, the server region attaches either CSMI, CPMI, or CVMI by default.

Specifying the TRANSID option with a blank transaction name is not valid. The error response in this case depends on how the link request is handled:

- If the linked-to program is defined locally, the link request succeeds in this situation.
- If the request is statically routed to a remote region, an INVREQ response is returned to the program initiating the link.
- If the request is dynamically routed to a remote region, a valid connection ID is returned, but after checking the TRANSID, CICS calls the dynamic routing program again with the parameter DYRFUNC set to 1 and the parameter DYRERROR set to 8. If the dynamic routing program responds by running the program locally, the link request succeeds. If the dynamic routing program responds by retrying without success, the link request is rejected, and a PGMIDERR response is returned to the program initiating the link.

Conditions

122 CHANNELERR

RESP2 values:

1 The name specified on the CHANNEL option contains an illegal character or combination of characters.

16 INVREQ

RESP2 values:

8 A LINK command with the INPUTMSG option is issued for a program that is not associated with a terminal, or that is associated with an APPC logical unit, or an IRC session.

14 The SYNCONRETURN option is specified but the program issuing the link request (the client program) is already in conversation with a mirror task in the remote region specified on the SYSID option. That is, a unit of work (UOW) is in progress, or the system initialization parameter **MROFSE=YES** is specified in the client region, or the MIRRORLIFE IPCONN setting is specified as task or UOW in the remote region. In this case, the client program is in an incorrect state to support the SYNCONRETURN option.

15 The program issuing the link request is already in conversation with a mirror task and the TRANSID specified is different from the transaction identifier of the active mirror.

16 The TRANSID specified is all blanks.

17 The TRANSID supplied by the dynamic routing program is all blanks.

19 A LINK command with the INPUTMSG option is issued for a program that is the subject of a DPL request; that is, SYSID is also specified.

30 The program manager domain has not yet been initialized, probably because a link request was made in a first stage PLT.

48 A LINK has been attempted to a Java program, but the user class cannot be found.

51 A LINK has been attempted to a Java program, but the JVMSERVER resource cannot be found.

52 A LINK has been attempted to a Java program, but the JVMSERVER resource is not enabled.

53 A LINK has been attempted to an application in a Liberty JVM server, but the LINK listener process was not available. The most likely cause of this error is that the feature `cicsts:link-1.0` is not enabled. To resolve this, add this feature to the `server.xml`.

54 The COMMAREA option is specified, but it is invalid when linking to an application in a Liberty JVM server.

55 The INPUTMSG option is specified, but it is invalid when linking to an application in a Liberty JVM server.

Default action: end the task abnormally.

Note: RESP2 values are not returned to the client for conditions occurring in a DPL server program.

22 LENGERR

RESP2 values:

11 The COMMAREA length is less than 0 or greater than the permitted length.

12 The length specified on the DATALENGTH option is a negative value.

13 The length specified on the DATALENGTH option is greater than the length specified on the LENGTH option.

26 The COMMAREA address is zero, but the COMMAREA length is nonzero.

27 The INPUTMSG length is less than 0 or greater than 32767.

Also occurs (RESP2 not set) in any of the following situations:

- The length specified on the LENGTH option is greater than the length of the data area specified in the COMMAREA option, and while that data was being copied a destructive overlap occurred because of the incorrect length.

Default action: end the task abnormally.

Note: RESP2 values are not returned to the client for conditions occurring in a DPL server program.

70 NOTAUTH

RESP2 values:

101 A resource security check has failed on PROGRAM(name).

Default action: end the task abnormally.

27 PGMIDERR

RESP2 values:

1 A program has no installed resource definition and either program autoinstall was switched off or the program autoinstall control program indicated that the program should not be autoinstalled.

This RESP2 can also occur if the program is a private program for an application, which is not defined as entry point, and the tasks current application context does not permit it to be called.

2 A program is disabled.

3 A program was not loaded because

- This load of the program was the first one and the program load failed, typically because the load module was not found.
- This load of the program was a subsequent load, but the first load failed.

To reset the load status, the load module must be in the DFHRPL concatenation, and a SET PROGRAM NEWCOPY is required.

21 The program autoinstall control program failed either because the program autoinstall control program is incorrect, incorrectly defined, or as a result of an abend in the program autoinstall control program. Program autoinstall is disabled and message DFHPG0202 or DFHPG0203 is written to the CSPL.

22 The model returned by the program autoinstall control program was not defined to CICS or was not enabled.

23 The program autoinstall control program returned invalid data.

24 Define for the program failed because autoinstall returned an invalid program name or definition.

25 The dynamic routing program rejected the link request.

26

A LINK has been attempted to an application in a Liberty JVM server, but the request has failed and the target application was not invoked. Message DFHSJ1207 gives more details about why the application could not be invoked.

27 A LINK has been attempted to an application in a Liberty JVM server, but the request timed out and the target application was not invoked. For more information, see Troubleshooting Liberty JVM servers and Java web applications.

Default action: end the task abnormally.

Note: RESP2 values are not returned to the client for conditions occurring in a DPL server program.

121 RESUNAVAIL

RESP2 values:

0 A resource required by the linked-to program is unavailable on the target region. The RESUNAVAIL condition applies to dynamically routed distributed program link (DPL) requests.

RESUNAVAIL is returned on the **EXEC CICS LINK** command *executed by the mirror in the target region*, if an XPCERES global user exit program indicates that a required resource is unavailable on the target region. It is not returned to the application.

Default action: call again the dynamic routing program for route selection failure.

82 ROLLEDBACK

RESP2 values:

29 The SYNCONRETURN option is specified and the server program cannot successfully take a sync point. The server program has taken a rollback, and all changes made to recoverable resources in the remote region, in the current unit of work, are backed out.

Default action: end the task abnormally.

53 SYSIDERR

RESP2 values:

18 The SYSID specified cannot be found in the intersystem table.

20 The remote system specified by SYSID is an LUTYPE6.1-connected system. Distributed program link requests are not supported on LUTYPE6.1 connections.

Note:

1. No local queuing takes place in the event of a SYSIDERR.
2. RESP2 values are not returned for conditions occurring on DPL requests.

21 The CHANNEL option was used and the LINK request was shipped or routed to a remote system which does not support it. (IPIC and MRO connections.)

28 The remote system specified by SYSID is not in service. This response can also indicate that the transaction has not been defined on the remote system.

29 The remote system specified by SYSID is in service, but there are no sessions available, and the dynamic routing program has chosen not to queue the link request.

31 The request to allocate a session to the remote system has been rejected.

32 The queue of allocate requests for sessions to the remote system has failed because the session allocation queue is full or has been purged.

Default action: end the task abnormally.

81 TERMERR
RESP2 values:

17 An unrecoverable error occurs during the conversation with the mirror; for example, if the session fails, or if the server region fails.

Default action: end the task abnormally.

If the SYNCONRETURN option was not specified on the LINK, the client program must decide whether to abend or roll back on receipt of this condition.

Note: RESP2 values are not returned to the client for conditions occurring in a DPL server program.

Examples

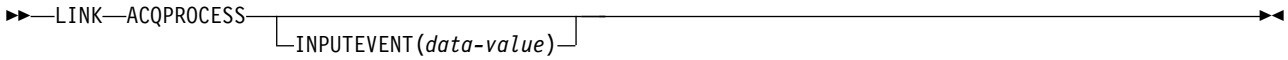
The following example shows how to request a link to an application program called PROGNAME:

```
EXEC CICS LINK PROGRAM(PROGNAME)
               COMMAREA(COMA) LENGTH(LENA)
               DATALENGTH(LENI) SYSID('CONX')
```

LINK ACQPROCESS

Execute a CICS business transaction services process synchronously without context-switching.

LINK ACQPROCESS



Conditions: EVENTERR, INVREQ, IOERR, NOTAUTH, PGMIDERR, PROCESSBUSY, PROCESSERR

Description

LINK ACQPROCESS executes the CICS business transaction services process currently acquired by the requestor. The process is executed synchronously with the requestor, with no context-switching.

The only process that a program can link is the one that it has acquired in the current unit of work. (Note, however, that if the program is running as the activation of an activity, it must use a RUN, not a LINK, command to activate the process it has acquired.) See Acquiring processes and activities .

To check the response from the process, the CHECK ACQPROCESS command must be used. This is because the response to the request to activate the process does not contain any information about the success or failure of the process itself—only about the success or failure of the request to activate it. Typically, the CHECK command is issued immediately after the LINK command.

LINK ACQPROCESS causes BTS to invoke the process's root activity and send it an input event. If the root activity is in its initial state—that is, if this is the first time it is to be run—CICS sends it the DFHINITIAL system event. If the root activity is not in its initial state, the input event must be specified on the INPUTEVENT option.

No context-switching

When an process is activated by a LINK ACQPROCESS command, it is invoked synchronously with the requestor and:

- In the same unit of work as the requestor
- With the transaction attributes (TRANSID and USERID) of the requesting transaction.

In other words, there is no context-switch. To invoke a process synchronously *with* context-switching—that is, in a separate UOW from that of the requesting transaction and with the TRANSID and USERID attributes specified on its DEFINE PROCESS command—use the RUN ACQPROCESS SYNCHRONOUS command.

Note: A context-switch always occurs when a process is run asynchronously.

If performance is more important than failure isolation, recoverability, and security, use LINK ACQPROCESS rather than RUN ACQPROCESS SYNCHRONOUS.

Options

ACQPROCESS
specifies that the process currently acquired by the requestor is to be run.

INPUTEVENT(data-value)
specifies the name (1–16 characters) of the event that causes the process to be attached.

You *must not* specify this option if the process's root activity is in its initial state; that is, if this is the first time the process is to be run. In this case, CICS sends the root activity the DFHINITIAL system event.

You *must* specify this option if the root activity is not in its initial state; that is, if it has been activated before.

If you specify INPUTEVENT, for the LINK command to be successful the root activity must have defined the named event as an input event.

Conditions

111 EVENTERR
RESP2 values:

7 The event named on the INPUTEVENT option has not been defined by the root activity of the process to be run as an input event; or its fire status is FIRED.

16 INVREQ
RESP2 values:

15 The task that issued the LINK command has not defined or acquired a process.

23 The process is suspended, and therefore cannot be run synchronously.

40 The program that implements the process to be run is remote.

44 A LINK has been attempted to a Java program, but the JVM pool is disabled.

45 A LINK has been attempted to a Java program, but the JVM profile cannot be found.

46 A LINK has been attempted to a Java program, but the JVM profile is not valid.

47 A LINK has been attempted to a Java program, but the system properties file cannot be found.

48 A LINK has been attempted to a Java program, but the user class cannot be found.

49 The shared class cache is STOPPED and autostart is disabled, so a Java program requesting use of the shared class cache cannot be executed.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to run the process.

27 PGMIDERR
RESP2 values:

1 A program has no installed resource definition and either program autoinstall was switched off, or the program autoinstall user program indicated that the program should not be autoinstalled.

2 A program is disabled.

3 A program could not be loaded because:

- This was the first load of the program and the program load failed, usually because the load module could not be found.
- This was a subsequent load of the program, but the first load failed.

In order to reset the load status the load module must be in the DFHRPL or dynamic LIBRARY concatenation, and a SET PROGRAM NEWCOPY will be required.

21 The program autoinstall user program failed either because the program autoinstall user program is incorrect, incorrectly defined, or as

a result of an abend in the program autoinstall user program. Program autoinstall is disabled and message DFHPG0202 or DFHPG0203 written to the CSPL.

22 The model returned by the program autoinstall user program was not defined to CICS, or was not enabled.

23 The program autoinstall user program returned invalid data.

24 Define for the program failed due to autoinstall returning an invalid program name or definition.

106 PROCESSBUSY
RESP2 values:

13 The request timed out. It may be that another task using this process-record has been prevented from ending.

108 PROCESSERR
RESP2 values:

6 Another process is current. That is, the program that issued the LINK command cannot link to the process it has acquired because it is itself running as an activation of a process.

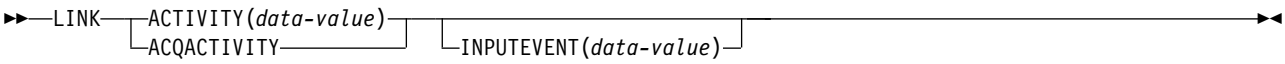
9 The process-type could not be found.

14 The root activity of the process to be run is not in INITIAL or DORMANT mode.

LINK ACTIVITY

Execute a CICS business transaction services activity synchronously without context-switching.

LINK ACTIVITY



Conditions: ACTIVITYBUSY, ACTIVITYERR, EVENTERR, INVREQ, IOERR, LOCKED, NOTAUTH, PGMIDERR

Description

LINK ACTIVITY executes a CICS business transaction services activity synchronously with the requestor, with no context-switching. The activity must previously have been defined to BTS.

LINK ACTIVITY causes BTS to invoke the activity and send it an input event. If the activity is in its initial state—that is, if this is the first time it is to be run, or if it has been reset by a RESET ACTIVITY command—CICS sends it the DFHINITIAL system event. If the activity is not in its initial state, the input event must be specified on the INPUTEVENT option.

The only activities a program can link to are as follows:

- If it is running as the activation of an activity, its own child activities. It can link to several of its child activities within the same unit of work.

- The activity it has acquired, by means of an ACQUIRE ACTIVITYID command, in the current unit of work. (Note, however, that if the program is running as the activation of an activity, it must use a RUN, not a LINK, command to activate the activity it has acquired.)

To check the response from the activity, the CHECK ACTIVITY command must be used. This is because the response to the request to activate the activity does not contain any information about the success or failure of the activity itself—only about the success or failure of the request to activate it. Typically, the CHECK command is issued immediately after the LINK command.

No context-switching

When an activity is activated by a LINK ACTIVITY command, it is invoked synchronously with the requestor and:

- In the same unit of work as the requestor
- With the transaction attributes (TRANSID and USERID) of the requesting transaction.

In other words, there is no **context-switch**. To invoke an activity synchronously *with* context-switching—that is, in a separate UOW from that of the requesting transaction and with the TRANSID and USERID attributes specified on its DEFINE ACTIVITY command—use the RUN ACTIVITY SYNCHRONOUS command.

Note: A context-switch always occurs when an activity is run asynchronously.

If performance is more important than failure isolation, recoverability, and security, use LINK ACTIVITY rather than RUN ACTIVITY SYNCHRONOUS.

Options

ACQACTIVITY
specifies that the activity to be run is the one that the current unit of work has acquired by means of an ACQUIRE ACTIVITYID command.

ACTIVITY(data-value)
specifies the name (1–16 characters) of the activity to be run. The name must be that of a child of the current activity.

INPUTEVENT(data-value)
specifies the name (1–16 characters) of the event that causes the activity to be attached.

You *must not* specify this option if the activity is in its initial state; that is, if this is the first time it is to be run, or if it has been reset by a RESET ACTIVITY command. In this case, CICS sends the activity the DFHINITIAL system event.

You *must* specify this option if the activity is not in its initial state; that is, if it has been activated before, and has not been reset by a RESET ACTIVITY command.

If you specify INPUTEVENT, for the LINK command to be successful the activity to be attached must have defined the named event as an input event.

Conditions

107 ACTIVITYBUSY
RESP2 values:

19 The request timed out. It may be that another task using this activity-record has been prevented from ending.

109 ACTIVITYERR
RESP2 values:

8 The activity named on the ACTIVITY option could not be found.

14 The target activity is not in the correct mode to process the specified event option. If the INPUTEVENT option was not specified, the activity must be in INITIAL mode. If the INPUTEVENT option was specified, the activity must be in DORMANT mode.

111 EVENTERR
RESP2 values:

7 The event named on the INPUTEVENT option has not been defined by the activity to be run as an input event; or its fire status is FIRED.

16 INVREQ
RESP2 values:

4 The ACTIVITY option was used to name a child activity, but the command was issued outside the scope of a currently-active activity.

21 The activity is suspended, and therefore cannot be run synchronously.

24 The ACQACTIVITY option was used, but the issuing task has not acquired an activity.

40 The program that implements the activity is remote.

44 A LINK has been attempted to a Java program, but the JVM pool is disabled.

45 A LINK has been attempted to a Java program, but the JVM profile cannot be found.

46 A LINK has been attempted to a Java program, but the JVM profile is not valid.

47 A LINK has been attempted to a Java program, but the system properties file cannot be found.

48 A LINK has been attempted to a Java program, but the user class cannot be found.

49 The shared class cache is STOPPED and autostart is disabled, so a Java program requesting use of the shared class cache cannot be executed.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to run the activity.

27 PGMIDERR

RESP2 values:

1 A program has no installed resource definition and either program autoinstall was switched off, or the program autoinstall user program indicated that the program should not be autoinstalled.

2 A program is disabled.

3 A program could not be loaded because:

- This was the first load of the program and the program load failed, usually because the load module could not be found.
- This was a subsequent load of the program, but the first load failed.

In order to reset the load status the load module must be in the DFHRPL or dynamic LIBRARY concatenation, and a SET PROGRAM NEWCOPY will be required.

21 The program autoinstall user program failed either because the program autoinstall user program is incorrect, incorrectly defined, or as a result of an abend in the program autoinstall user program. Program autoinstall is disabled and message DFHPG0202 or DFHPG0203 written to the CSPL.

22 The model returned by the program autoinstall user program was not defined to CICS, or was not enabled.

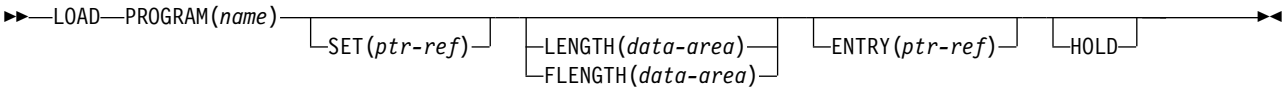
23 The program autoinstall user program returned invalid data.

24 Define for the program failed due to autoinstall returning an invalid program name or definition.

LOAD

Load a program from the CICS DFHRPL or dynamic LIBRARY concatenation into main storage.

LOAD



Conditions: INVREQ, LENGERR, NOTAUTH, PGMIDERR

This command is threadsafe.

Note for dynamic transaction routing: Using LOAD with HOLD, or using a resource that has been defined with RELOAD=YES, could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. For more information about transaction affinities, see Affinity .

Description

Load makes a copy of an application program, table, or map available to the invoking task. If the program is defined with RELOAD=NO, it is fetched from the library concatenation where it resides only if there is not a copy already in main storage. If the program is defined with RELOAD=YES, a new copy is always

fetched from the LIBRARY concatenation. (See for further details about maps.) Using LOAD can reduce system overhead.

Options

ENTRY(*ptr-ref***)**
Specifies the pointer reference to be set to the address of the entry point in the program that has been loaded. CICS program load services set the entry point according to the addressing mode of the load module:

- AMODE(24): bit 0 is 0 and bit 31 is 0.
- AMODE(31): bit 0 is 1 and bit 31 is 0.
- AMODE(64): bit 0 is 0 and bit 31 is 1.

For assembler programs without an explicit ENTRY defined in the linkedit definitions, the entry point returned depends on whether there is a CICS stub, and whether the LOAD command is issued from a PLT program:

- If there is a CICS stub, the entry point address is incremented for this stub unless the LOAD command is issued from a PLT program run during the first phase of initialization or the final phase of shutdown.
- If there is not a CICS stub, the entry point address is the same as the load point address.

FLENGTH(*data-area***)**
Specifies a fullword binary area to be set to the length of the loaded program, table, or map. Use FLENGTH if the length of the loaded program is greater than 32 KB.

HOLD
Specifies that the loaded program, table, or map is not to be released (if still available) when the task that issues the LOAD command is terminated; it is to be released only in response to a RELEASE command from this task or from another task.

If you omit HOLD, the program, table, or map is released when the task that issued the load terminates or issues a RELEASE command.

If, however, the program is defined with RELOAD=YES, neither of the above apply. RELEASE does not work, and a FREEMAIN request must be issued to remove the program.

LENGTH(*data-area***)**
Specifies a halfword binary value to be set to the length of the loaded program, table, or map. To avoid raising the LENGERR condition, use FLENGTH if the length of the loaded program is likely to be greater than 32 KB.

PROGRAM(*name***)**
Specifies the identifier (1 - 8 characters) of a program, table, or map to be loaded. The specified name must have been defined as a program to CICS, though if AUTOINSTALL is active, a definition is autoinstalled.

SET(*ptr-ref***)**
Specifies the pointer reference that is to be set to the address at which a program, table, or map is loaded.

Conditions

16 INVREQ
RESP2 values:

30 The program manager domain is not yet initialized. This is probably because a load request was made in a first stage PLT.

Default action: terminate the task abnormally.

22 LENGERR
RESP2 values:

19 LENGTH is used and the length of the loaded program is greater than 32 KB.

Default action: terminate the task abnormally.

70 NOTAUTH
RESP2 values:

101 A resource security check has failed on PROGRAM(name).

Default action: terminate the task abnormally.

27 PGMIDERR
RESP2 values:

1 A program, table, or map has no installed resource definition and either program autoinstall was switched off, or the program autoinstall control program indicated that the program should not be autoinstalled.

2 A program is disabled.

3 A program could not be loaded for one of the following reasons:

- This was the first load of the program and the program load failed, usually because the load module could not be found.
- This was a subsequent load of the program, but the first load failed.

To reset the load status, the load module must be in the DFHRPL or dynamic LIBRARY concatenation, and a SET PROGRAM NEWCOPY will be required.

9 The installed program definition is for a remote program.

21 The program autoinstall control program failed either because the program autoinstall control program is incorrect, or incorrectly defined, or as a result of an abend in the program autoinstall control program. Program autoinstall is disabled and message DFHPG0202 or DFHPG0203 written to the CSPL.

22 The model returned by the program autoinstall control program was not defined in CICS, or was not enabled.

23 The program autoinstall control program returned invalid data.

24 Define for the program failed because autoinstall returned an invalid program name or definition.

42 An attempt has been made to LOAD a JVM program. This action is not valid because Java byte code programs are not managed by CICS Loader.

Default action: terminate the task abnormally.

Example

The following example shows how to load a user-prepared table called TB1:

```
EXEC CICS LOAD PROGRAM('TB1') SET(PTR)
```

MONITOR

Code a user event-monitoring point.

MONITOR



Condition: INVREQ

This command is threadsafe.

Description

MONITOR provides information about the performance of your application transactions. It replaces the monitoring aspects of ENTER TRACEID.

In addition to the monitoring data collected at predefined event monitoring points (EMPs) in CICS, a user application program can contribute data to user fields in the CICS monitoring records. You can do this by using the MONITOR command to invoke user-defined EMPs. At each of these user EMPs, you can add or change 1 - 16384 bytes of your own data in each performance monitoring record. In those 16384 bytes, you can have any combination of the following:

- 0 through 256 counters
- 0 through 256 clocks
- A single 8192-byte character string

Options

DATA1(*data-area***)**
Specifies a 4-byte variable whose contents depend on the type of user EMP being used:

- If the user EMP contains an ADDCNT, SUBCNT, NACNT, EXCNT, or ORCNT option, the DATA1 variable is an area used as defined by the MCT user EMP definition.
- If the MCT user EMP definition contains an MLTCNT option, the DATA1 variable is an area with the address of a series of adjacent fullwords containing the values to be added to the user count fields defined in the MCT user EMP definition.
- If the MCT user EMP definition contains a MOVE option, the DATA1 variable is an area with the address of the character string to be moved.

See User event monitoring points - DFHMCT TYPE=EMP for details of user EMP options.

DATA2(*data-area***)**
Specifies a 4-byte variable whose contents depend on the type of user EMP being used:

- If the user EMP contains an ADDCNT, SUBCNT, NACNT, EXCNT, or ORCNT option, the DATA2 variable is an area used as defined by the MCT user EMP definition.
- If the MCT user EMP definition contains an MLTCNT option, the DATA2 variable is an area with the number of user count fields to be updated. The

number specified in DATA2 overrides the default value defined in the MCT for the operation. The default value depends on the option that you have defined in the EMP definition. If you specify a null value in DATA2, monitoring uses the default value that is specified in the EMP definition. If DATA2 is not specified, the MLTCNT operation raises an INVREQ condition although the operation was successful.

- If the MCT user EMP definition contains a MOVE option, the DATA2 variable is an area with the length of the character string to be moved. The number specified in DATA2 will override the default value defined in the MCT for the operation. The default value depends on the option that you have defined in the EMP definition. If you specify a null value in DATA2, monitoring uses the default value that is specified in the EMP definition. If DATA2 is not specified, the MOVE operation raises an INVREQ although the operation was successful.

Examples of invoking application naming EMPs provides an example of how the default value is handled for EMP.

See User event monitoring points - DFHMCT TYPE=EMP for details of user EMP options.

ENTRYNAME(*data-area***)**
Is the monitoring point entry name that qualifies the POINT value and is defined in the monitoring control table (MCT). ENTRYNAME defaults to USER if not specified. Specify in the data-area the name of the 8-byte field in your application program that contains the monitoring point entry name.

POINT(*data-value***)**
Specifies the monitoring point identifier as defined in the MCT, and is in the range 0 - 255. Note, however, that point identifiers in the range 200 - 255 are reserved for use by IBM program products.

Conditions

16 INVREQ
RESP2 values:

1 Your POINT value is outside the range 1 through 255.

2 Your POINT value is not defined in the MCT.

3 Your DATA1 value is not valid.

4 Your DATA2 value is not valid.

5 You did not specify DATA1 for an MCT operation that required it.

6 You did not specify DATA2 for an MCT operation that required it.

Default action: terminate the task abnormally.

Examples

For example, you could use these user EMPs to count the number of times a certain event occurs, or to time the interval between two events.

Figure 2 on page 343 gives examples of MONITOR commands (and of the MCT entries you need for them).

Note:

1. Example 1 shows a user clock being started by an application identified as PROG3. This is the eleventh EMP in this application. To prevent confusion with the eleventh EMP in another application, this EMP is uniquely identified by the tag ENTRY3.11. The clock that is being started is the first clock in a string.
2. Example 2 shows the same user clock being stopped, by the same application, but from a different EMP. The EMP is uniquely identified by the tag ENTRY3.12.
3. Example 3 shows some user data being loaded into the 32-byte character string reserved for that purpose. The loading starts at offset 0, and the data is no more than 32 bytes in length.

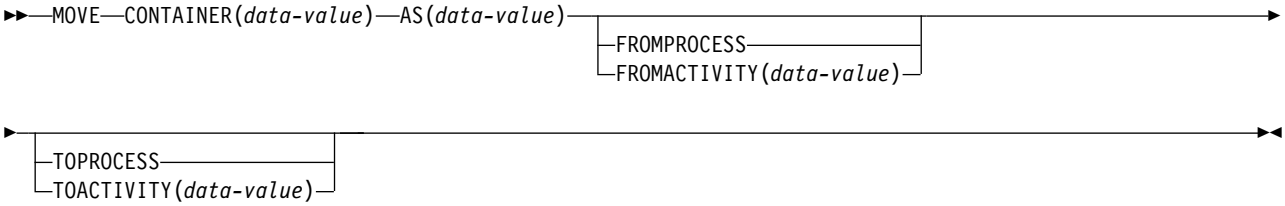
```
1:
EXEC CICS MONITOR
          POINT(11)
          ENTRYNAME(ENTRY3)
  needing:  DFHMCT TYPE=EMP,
                   CLASS=PERFORM,
                   ID=(ENTRY3.11),
                   CLOCK=(1,CLOCKA),
                   PERFORM=SCLOCK(1)
2:
EXEC CICS MONITOR
          POINT(12)
          ENTRYNAME(ENTRY3)
  needing:  DFHMCT TYPE=EMP,
                   CLASS=PERFORM,
                   ID=(ENTRY3.12),
                   PERFORM=PCLOCK(1)
3:
EXEC CICS MONITOR
          POINT(13)
          DATA1(address of data)
          DATA2(length of data)
          ENTRYNAME(ENTRY3)
  needing:  DFHMCT TYPE=EMP,
                   CLASS=PERFORM,
                   ID=(ENTRY3.13),
                   PERFORM=MOVE(0,32)
```

Figure 2. Examples of coding user EMPs

MOVE CONTAINER (BTS)

Move a BTS data-container (and its contents) from one activity to another.

MOVE CONTAINER (BTS)



Conditions: ACTIVITYERR, CONTAINERERR, INVREQ, IOERR, LOCKED

Description

MOVE CONTAINER (BTS) moves a data-container (and its contents) from one BTS activity to another. After the move, the source container is destroyed.

The source and target containers are identified by name and by the activities that own them. The activity that owns the source container can be identified:

- Explicitly, by specifying the FROMPROCESS or FROMACTIVITY option.
- Implicitly, by omitting the FROMPROCESS and FROMACTIVITY options. If these are omitted, the current activity is implied.

Similarly, the activity that owns the target container can be identified:

- Explicitly, by specifying the TOPROCESS or TOACTIVITY option.
- Implicitly, by omitting the TOPROCESS and TOACTIVITY options. If these are omitted, the current activity is implied.

You can move a container:

- From the current activity to a child of the current activity
- From a child of the current activity to the current activity
- From the current activity to the current activity (thus renaming the container)
- From one child of the current activity to another

In addition, *if the current activity is the root activity*, you can move a container:

- From the current process to the current (root) activity
- From the current process to a child of the current activity
- From the current process to the current process (thus renaming the container)
- From the current activity to the current process
- From a child of the current activity to the current process

You can use MOVE CONTAINER, instead of GET CONTAINER and PUT CONTAINER, as a more efficient way of transferring data between activities—for an explanation, see Container commands .

Note:

1. If the source container does not exist, an error occurs.
2. If the target container does not already exist, it is created. If the target container already exists, its previous contents are overwritten.
3. You cannot move containers from one process to another. Both the source and target containers must be within the scope of the current process.
4. Only the root activity can specify a process-container as the source or target of a MOVE CONTAINER command.

   A process's containers are *not* the same as its root activity's containers.

See also “GET CONTAINER (BTS)” on page 232 and “PUT CONTAINER (BTS)” on page 354.

Options

AS(data-value)
specifies the name (1–16 characters) of the target container. If the target container already exists, its contents are overwritten.

CONTAINER(data-value)
specifies the name (1–16 characters) of the source container that is to be moved.

FROMACTIVITY(data-value)
specifies the name (1–16 characters) of the activity that owns the source container. If specified, this option must name a child of the current activity (or the current activity itself).

FROMPROCESS
specifies that the source container is owned by the current process—that is, the process that the program that issues the command is executing on behalf of.

TOACTIVITY(data-value)
specifies the name (1–16 characters) of the activity that owns the target container. If specified, this option must name a child of the current activity (or the current activity itself).

TOPROCESS
specifies that the target container is owned by the current process—that is, the process that the program that issues the command is executing on behalf of.

Conditions

109 ACTIVITYERR
RESP2 values:

8 The activity named on the FROMACTIVITY or TOACTIVITY option could not be found.

110 CONTAINERERR
RESP2 values:

10 The container named on the CONTAINER option could not be found.

26 The process container named on the CONTAINER option is read-only.

16 INVREQ
RESP2 values:

4 The command was issued outside the scope of a currently-active activity.

25 The FROMPROCESS or TOPROCESS option was used, but the command was issued outside the scope of a currently-active process.

17 IOERR
RESP2 values:

30 An input/output error has occurred on the repository file.

31 The record on the repository file is in use.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

MOVE CONTAINER (CHANNEL)

Move a container (and its contents) from one channel to another.

MOVE CONTAINER (CHANNEL)



Conditions: CHANNELERR, CONTAINERERR, INVREQ

This command is threadsafe.

Description

MOVE CONTAINER (CHANNEL) moves a container from one channel to another. After the move, the source container no longer exists.

The source and target containers are identified by name and by the channels that own them. The channel that owns the source container can be identified as follows:

- Explicitly, by specifying the CHANNEL option.
- Implicitly, by omitting the CHANNEL option. If this is omitted, the current channel is implied.

Similarly, the channel that owns the target container can be identified as follows:

- Explicitly, by specifying the TOCHANNEL option.
- Implicitly, by omitting the TOCHANNEL option. If this is omitted, the current channel is implied.

You can move a container:

- From one channel to another.
- Within the same channel—for example, from the current channel to the current channel. This has the effect of renaming the container.

You can use **MOVE CONTAINER**, instead of **GET CONTAINER** and **PUT CONTAINER**, as a more efficient way of transferring data between channels.

Note:

1. The source channel must be within the scope of the program that issues the **MOVE CONTAINER** command.
2. If the target channel does not exist, within the scope of the program that issues the **MOVE CONTAINER** command, it is created.
3. If the source container does not exist, an error occurs.
4. If the target container does not already exist, it is created. If the target container already exists, its previous contents are overwritten.
5. If you try to overwrite a container with itself, nothing happens. That is, if you specify the same value for the CONTAINER and AS options, and either omit both the CHANNEL and TOCHANNEL options or give them the same value, so that the same channel is specified, the source container is not changed and not deleted. No error condition is raised.

Options

AS(data-value)
Specifies the name (1–16 characters) of the target container. If the target container already exists, its contents are overwritten.

The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = ¬ , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Container names are always in EBCDIC. The allowable set of characters for container names, listed above, includes some characters that do not have the same representation in all EBCDIC code pages. We therefore recommend that, if containers are to be shipped between regions, the characters used in naming them should be restricted to A-Z a-z 0-9 & : = , ; < > . - and _.

CHANNEL(data-value)
Specifies the name (1–16 characters) of the channel that owns the source container. You can specify the channel name DFHTRANSACTION to use the transaction channel. If this option is not specified, the current channel is implied.

CONTAINER(data-value)
Specifies the name (1–16 characters) of the source container that is to be moved.

TOCHANNEL(data-value)
Specifies the name (1–16 characters) of the channel that owns the target container. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters. If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

Channel names are always in EBCDIC. The set of allowed characters for channel names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be shipped between regions, it is advisable to restrict the characters that are used to name them to A-Z a-z 0-9 & : = , ; < > . - and _.

You can specify the channel name DFHTRANSACTION to use a transaction channel. A transaction channel does not go out of scope when the link level changes; it is always accessible in the transaction. For more information, see Channels and containers.

If this option is not specified, the current channel is implied.

Conditions

122 CHANNELERR
RESP2 values:

1 The name specified on the TOCHANNEL option contains an illegal character or combination of characters.

2 The channel specified on the CHANNEL option could not be found.

3 Either the current channel or the channel specified on the CHANNEL option is read-only.

110 CONTAINERERR
RESP2 values:

10 The container named on the CONTAINER option could not be found.

18 The name specified on the AS option contains an illegal character or combination of characters.

16 INVREQ
RESP2 values:

4 The CHANNEL or TOCHANNEL option (or both) was not specified, there is no current channel (because the program that issued the command was not passed one), and the command was issued outside the scope of a currently-active BTS activity.

30 You cannot move a CICS-defined read-only container.

31 You cannot move a container to (that is, overwrite) an existing, CICS-defined, read-only container.

POINT

Get information about an LUTYPE6.1 logical unit.

POINT



Condition: NOTALLOC

Description

POINT gets information about a named facility, such as whether it owns the given facility.

This command can be used on an MRO session.

Options

CONVID(*name*)
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

SESSION(*name*)
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used. If both this option and CONVID are omitted, the principal facility for the task is used.

Conditions

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

POP HANDLE

Restore the stack.

POP HANDLE



Condition: INVREQ

This command is threadsafe.

Description

Use the **POP HANDLE** command to restore the effect of **IGNORE CONDITION**, **HANDLE ABEND**, **HANDLE AID**, and **HANDLE CONDITION** commands to the state they were in before a **PUSH HANDLE** command was executed at the current link level.

Restriction: This command is supported only in COBOL, PL/I, and assembler language applications (but not AMODE(64) assembler language applications). It is not supported in all other supported high level languages.

This command can be useful, for example, during a branch to a subroutine embedded in a main program.

Normally, when a CICS program calls a subroutine (at the same logical level), the program or routine that receives control inherits the current **HANDLE** commands. These commands might not be appropriate in the called program. The called program can use **PUSH HANDLE** to suspend existing **HANDLE** commands, and before returning control to the caller, can then restore the original commands using the **POP HANDLE** command.

Note: When a CICS program uses EXEC CICS LINK to call another CICS program, the HANDLE effects are not inherited by the linked-to program, but CICS will search preceding logical levels for a **HANDLE ABEND** exit. See Abnormal termination recovery for further details about the relationship between LINK and HANDLE ABEND.

You can nest **PUSH HANDLE ... POP HANDLE** command sequences within a task. Each POP HANDLE command restores a set of specifications.

Conditions

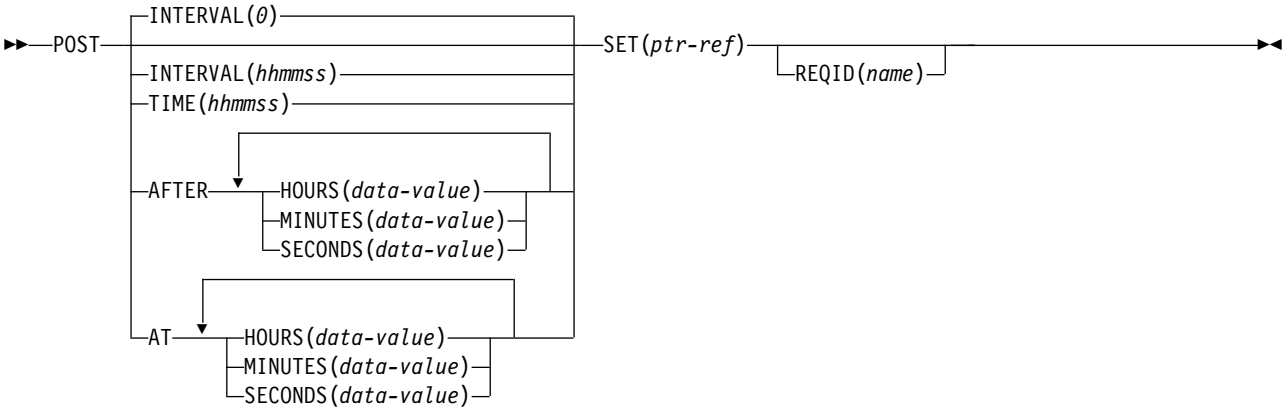
16 INVREQ
Occurs if no matching **PUSH HANDLE** command has been executed at the current link level.

Default action: terminate the task abnormally.

POST

Request notification when a specified time has expired.

POST



Conditions: EXPIRED, INVREQ

Note for dynamic transaction routing: Using POST if later CANCELed by another task could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. See Affinity for more information about transaction affinities.

Description

POST requests notification that a specified time has expired. In response to this command, CICS makes a timer-event control area available for testing. This 4-byte control area is initialized to binary zeros, and the pointer reference specified in the SET option is set to its address.

When the time you specify has expired, the timer-event control area is posted; that is, its first byte is set to X'40' and its third byte to X'80'. You can test posting in either of the following ways:

- By checking the timer-event control area at intervals. You must give CICS the opportunity to post the area; that is, the task must relinquish control of CICS before you test the area. Normally, this condition is satisfied as a result of other commands being issued; if a task is performing a long internal function, you can force control to be relinquished by issuing a SUSPEND command.
- By suspending task activity by a WAIT EVENT or WAIT EXTERNAL command until the timer-event control area is posted. This action is similar to issuing a DELAY command but, with a POST and WAIT EVENT or WAIT EXTERNAL command sequence, you can do some processing after issuing the POST command; a DELAY command suspends task activity at once. No other task should attempt to wait on the event set up by a POST command.
- By using WAITCICS.

The timer-event control area can be released for a variety of reasons. If this happens, the result of any other task issuing a WAIT command on the event set up by the POST command is unpredictable.

However, other tasks can cancel the event if they have access to the REQID associated with the POST command. (See the CANCEL command and the description of the REQID option.) A timer-event control area provided for a task is not released or altered (except as described above) until one of the following events occurs:

- The task issues a subsequent DELAY or POST command.
- The task issues a subsequent START command naming a transaction in the local system. (A START command that names a transaction on a remote system does not affect the event set up by the POST command, unless the transaction is defined with LOCALQ set to YES and local queuing is performed.)
- The task issues a CANCEL command to cancel the POST command.
- The task is terminated, normally or abnormally.
- Any other task issues a CANCEL command for the event set up by the POST command.

A task can have only one POST command active at any given time. Any DELAY or POST command, or a START command naming a transaction in the local system, supersedes a POST command previously issued by the task.

The default is INTERVAL(0), but for C the default is AFTER HOURS(0) MINUTES(0) SECONDS(0).

Options

AFTER
specifies the interval of time to elapse.

There are two ways to enter the time under AFTER and AT.

1. A combination of at least two of HOURS(0–99), MINUTES(0–59), and SECONDS(0–59). HOURS(1) SECONDS(3) would mean one hour and three seconds (the minutes default to zero).
2. As one of HOURS(0–99), MINUTES(0–5999), or SECONDS(0–359 999). HOURS(1) means one hour. MINUTES(62) means one hour and two minutes. SECONDS(3723) means one hour, two minutes, and three seconds.

AT
specifies the time of expiring. For the ways to enter the time, see the AFTER option.

HOURS(*data-value***)**
specifies a fullword binary value in the range 0–99. This is a suboption of the AFTER and AT options. For its use and meaning, see the AFTER option.

INTERVAL(*hhmmss***)**
specifies an interval of time that is to elapse from the time at which the POST command is issued. The **mm** and **ss** are in the range 0–59. The time specified is added to the current clock time by CICS when the command is executed to calculate the expiration time.

This option is used to specify when the posting of the timer-event control area should occur.

When using the C language, you are recommended to use the AFTER/AT HOURS, MINUTES, and SECONDS options as C does not provide a packed decimal data type. You may use INTERVAL, but if the value specified is **not** an integer constant, the application is responsible for ensuring that the value passed to CICS is in packed decimal format.

MINUTES(*data-value***)**
specifies a fullword binary value in the range 0–59, when HOURS or SECONDS are also specified, or 0–5999 when MINUTES is the only option specified. This is a suboption of the AFTER and AT options. For its use and meaning, see the AFTER option.

REQID(*name***)**
specifies a name (1–8 characters), which should be unique, to identify the POST request. Using this option to specify an application-defined name is one way to enable another transaction to cancel the POST request.

If you do not specify your own REQID, CICS generates a unique request identifier for you in the EIBREQID field of the EXEC interface block. This, like your own REQID, can be used by another transaction to cancel the POST request.

To enable other tasks to cancel unexpired POST requests, you must make the request identifier dynamically available. For example, storing it in a TS queue, whose name is known to other applications that may want to cancel the POST request, is one way you can pass a request identifier to other transactions.

SECONDS(*data-value***)**
specifies a fullword binary value in the range 0–59, when HOURS or MINUTES are also specified, or 0–359 999 when SECONDS is the only option specified. This is a suboption of the AFTER and AT options. For its use and meaning, see the AFTER option.

SET(*ptr-ref***)**
specifies the pointer reference to be set to the address of the 4-byte timer-event control area generated by CICS. This area is initialized to binary zeros; on expiration of the specified time, the first byte is set to X'40', and the third byte to X'80'.

The timer-event control area always resides below the 16MB line in shared dynamic storage (SDSA).

TIME(*hhmmss***)**
specifies the time when the posting of the timer-event control area should occur.

When using the C language, you are recommended to use the AFTER/AT HOURS, MINUTES, and SECONDS options as C does not provide a packed decimal data type. You may use TIME, but if the value specified is **not** an integer constant, the application is responsible for ensuring that the value passed to CICS is in packed decimal format. See the section about expiration times in Interval control.

Conditions

31 EXPIRED
occurs if the time specified has already expired when the command is issued.

Default action: ignore the condition.

16 INVREQ
RESP2 values:

4 Hours are out of range.

5 Minutes are out of range.

6 Seconds are out of range.

also occurs (RESP2 not set) in any of the following situations:

- The POST command is not valid for processing by CICS.

Default action: terminate the task abnormally.

Examples

The following example shows you how to request a timer-event control area for a task, to be posted after 30 seconds:

```
EXEC CICS POST
     INTERVAL(30)
     REQID('RBL3D')
     SET(PREF)
```

The following example shows you how to ask to be notified when the specified time of day is reached. Because no request identifier is specified in the command, CICS automatically assigns one and returns it to the application program in the EIBREQID field in the EIB.

```
EXEC CICS POST
     TIME(PACKTIME)
     SET(PREF)
```

PURGE MESSAGE

Discontinue building a BMS logical message.

PURGE MESSAGE



Conditions: Full BMS: INVREQ, TSIOERR

Description

PURGE MESSAGE discontinues the building of a BMS logical message. It deletes the current logical message, including any pages of device-dependent data stream already written to CICS temporary storage. The application program may then build a new logical message.

The portions of the logical message already built in main storage or in temporary storage are deleted.

See BMS macros for map definition macros.

PURGE MESSAGE is only available on full-function BMS. For further information about BMS, see Basic mapping support.

Conditions

16 INVREQ
RESP2 values:

200 The command was called in a distributed program link server program.

Default action: terminate the task abnormally.

35 TSIOERR
occurs if there is an irrecoverable temporary storage input/output error.

Default action: terminate the task abnormally.

PUSH HANDLE

Suspend the stack.

PUSH HANDLE



This command is threadsafe.

Description

You can use the **PUSH HANDLE** command to suspend the current effect of the **IGNORE CONDITION**, **HANDLE ABEND**, **HANDLE AID**, and **HANDLE CONDITION** commands.

Restriction: This command is supported only in COBOL, PL/I, and assembler language applications (but not AMODE(64) assembler language applications). It is not supported in all other supported high level languages.

This command can be useful, for example, during a branch to a subroutine embedded in a main program.

Normally, when a CICS program calls a subroutine at the same logical level, the program or routine that receives control inherits the current **HANDLE** commands. These commands may not be appropriate in the called program. The called program can use **PUSH HANDLE** to suspend existing **HANDLE** commands.

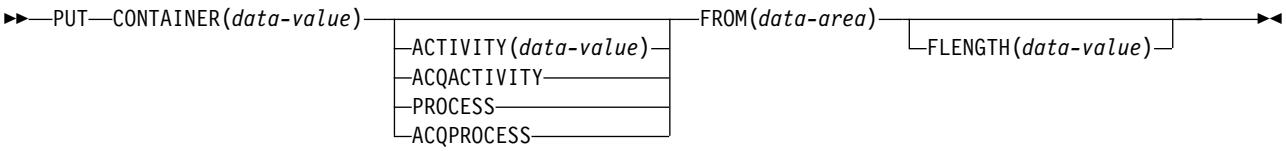
Note: When a CICS program uses EXEC CICS LINK to call another CICS program, the **HANDLE CONDITION** options are not inherited by the linked-to program, but CICS will search preceding logical levels for a **HANDLE ABEND** exit. See Abnormal termination recovery for further details about the relationship between LINK and HANDLE ABEND.

You can nest **PUSH HANDLE ... POP HANDLE** command sequences within a task. Each **PUSH HANDLE** command stacks a set of specifications.

PUT CONTAINER (BTS)

Save data in a named BTS data-container.

PUT CONTAINER (BTS)



Conditions: ACTIVITYERR, CONTAINERERR, INVREQ, IOERR, LOCKED, PROCESSBUSY

Description

PUT CONTAINER (BTS) saves data and places it in a container associated with a specified BTS activity or process.

The container is identified by name. The process or activity that owns the container can be identified:

- Explicitly, by specifying one of the PROCESS- or ACTIVITY-related options.
- Implicitly, by omitting the PROCESS- and ACTIVITY-related options. If these are omitted, the current activity is implied.

Note:

1. There is no limit to the number of containers that can be associated with an activity.
2. Different activities can own identically-named containers—these are different containers.
3. If the named container does not already exist, it is created. If the named container already exists, its previous contents are overwritten.
4. Containers owned by a process (*process-containers*) can be read by every activity in the process. However, they can be updated only by the root activity, or by a program that has acquired the process.

   A process's containers are *not* the same as its root activity's containers.

See also “GET CONTAINER (BTS)” on page 232 and “MOVE CONTAINER (BTS)” on page 343.

Options

ACQACTIVITY
specifies either of the following:

- If the program that issues the command has acquired a process, that the container is owned by the root activity of that process.
- Otherwise, that the container is owned by the activity that the program has acquired by means of an ACQUIRE ACTIVITYID command.

ACQPROCESS
specifies that the container is owned by the process that the program that issues the command has acquired in the current unit of work.

ACTIVITY(data-value)
specifies the name (1–16 characters) of the activity that owns the container. This must be a child of the current activity.

CONTAINER(data-value)
specifies the name (1–16 characters) of the container into which data is to be placed.

The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = ¬ , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

FLENGTH(data-value)
specifies, as a fullword binary value, the length of the data area from which data is to be read.

FROM(data-area)
specifies an area of working storage from which the data to be saved is to be read.

PROCESS
specifies that the container into which data is to be placed is owned by the current process—that is, the process that the program that issues the command is executing on behalf of.

Conditions

109 ACTIVITYERR
RESP2 values:

8 The activity named on the ACTIVITY option could not be found.

110 CONTAINERERR
RESP2 values:

10 The container named on the CONTAINER option could not be found.

18 The name specified on the CONTAINER option contains an illegal character or combination of characters.

26 The process container named on the CONTAINER option is read-only.

16 INVREQ
RESP2 values:

1 The DATATYPE option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one.) DATATYPE is valid only on PUT CONTAINER commands that specify (explicitly or implicitly) a channel. It is not valid on PUT CONTAINER (BTS) commands.

2 The FROMCCSID option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one.) FROMCCSID is valid only on PUT CONTAINER commands that specify (explicitly or implicitly) a channel. It is not valid on PUT CONTAINER (BTS) commands.

4 The command was issued outside the scope of a currently-active activity.

15 The ACQPROCESS option was used, but the unit of work that issued the request has not acquired a process.

24 The ACQACTIVITY option was used, but the unit of work that issued the request has not acquired an activity.

25 The PROCESS option was used, but the command was issued outside the scope of a currently-active process.

17 IOERR
RESP2 values:

30 An input/output error has occurred on the repository file.

31 The record on the repository file is in use.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

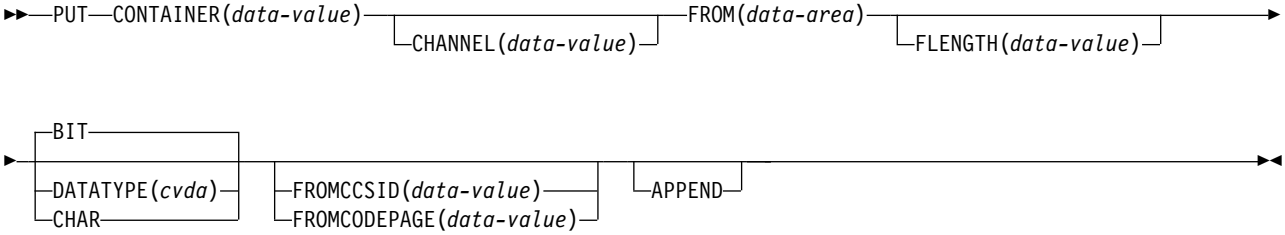
106 PROCESSBUSY
RESP2 values:

13 The request could not be satisfied because the process record is locked by another task.

PUT CONTAINER (CHANNEL)

Place data in a named channel container.

PUT CONTAINER (CHANNEL)



Conditions: CCSIDERR, CHANNELERR, CODEPAGEERR, CONTAINERERR, INVREQ, LENGERR

This command is threadsafe.

Description

PUT CONTAINER (CHANNEL) places data in a container associated with a specified channel.

The container is identified by name.

The channel that owns the container can be identified explicitly, by specifying the CHANNEL option, or implicitly, by omitting the CHANNEL option. When this option is omitted, the current channel is implied. If there is no current channel, the command is assumed to be an EXEC CICS PUT CONTAINER (BTS) command. For more information, see "PUT CONTAINER (BTS)" on page 354.

If the named container does not exist, it is created. If the named container exists, its previous contents are overwritten, unless you specify the APPEND option. If the named channel does not exist, it is created.

There is no limit to the number of containers that can be associated with a channel. The size of individual containers is limited only by the amount of storage available.

CAUTION:
If you create multiple large containers, you might limit the amount of storage available to other applications.

Options

APPEND
Specifies that the data passed to the container is appended to the existing data in the container. If this option is not stated, the existing data in the container is overwritten by the data passed to the container.

CHANNEL(*data-value*)
Specifies the name (1–16 characters) of the channel that owns the container. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Channel names are always in EBCDIC. The set of allowed characters for channel names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be shipped between regions, it is advisable to restrict the characters that are used to name them to A-Z a-z 0-9 & : = , ; < > . - and _.

If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

You can specify the channel name DFHTRANSACTION to use a transaction channel. A transaction channel does not go out of scope when the link level changes; it is always accessible in the transaction. For more information, see Channels and containers.

CONTAINER(*data-value*)
Specifies the name (1–16 characters) of the container into which data is placed.

The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Do not use container names that begin with DFH, unless requested to do so by CICS.

Container names are always in EBCDIC. The set of allowed characters for container names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if containers are to be shipped between regions, it is advisable to restrict the characters used to name themto A-Z 0-9 & : = , ; < > . - and _.

DATATYPE(*cvda*)
Specifies the type of data to put into the container. This option applies only to new containers. If the container exists, its data type was established when it was created and cannot be changed. CVDA values are:

BIT Bit data. The data in the container cannot be converted. This is the default value, unless FROMCCSID is specified.

CHAR
Character data. The data to store in the container is converted (if required) according to the setting in the FROMCCSID or FROMCODEPAGE value. If the FROMCCSID and FROMCODEPAGE options are not specified, it is assumed that the data is encoded in the CCSID of the region, as specified in the **LOCALCCSID** system initialization parameter.

All the data in a container is converted as if it were a single character string. For SBCS code pages, a structure consisting of several character fields is equivalent to a single-byte character string. However, for DBCS code pages this is not the case. If you use DBCS code pages, you must put each character string into a separate container to ensure that data conversion works correctly.

FLENGTH(*data-value*)
Specifies, as a fullword binary value, the length of the data area from which data is read.

FROM(*data-area*)
Specifies the data area from which the data is written to the container.

FROMCCSID(*data-value*)
Specifies the current Coded Character Set Identifier (CCSID) of the character data to put into the container, as a fullword binary number. If you prefer to specify an IANA name for the code page, or if you prefer to specify the CCSID as alphanumeric characters, use the FROMCODEPAGE option instead. Use this option if the data to place in the container is not encoded in the CCSID of the region, as specified in the LOCALCCSID system initialization parameter.

If the FROMCCSID option is specified, DATATYPE(DFHVALUE(CHAR)) is implied.

FROMCODEPAGE(*data-value*)
Specifies an IANA-registered alphanumeric charset name or a Coded Character Set Identifier (CCSID) for the current code page of the character data to put into the container, using up to 40 alphanumeric characters, including appropriate punctuation. Use this option instead of the CCSID option if you prefer to use an IANA-registered charset name, as specified in the Content-Type header for an HTTP request. CICS converts the IANA name into a CCSID, and the subsequent data conversion process is identical. Also use this option if you prefer to specify the CCSID in alphanumeric characters, rather than as a fullword binary number.

If the FROMCODEPAGE option is specified, DATATYPE(DFHVALUE(CHAR)) is implied.

Conditions

123 CCSIDERR
RESP2 values:

1 The CCSID specified on the FROMCCSID option is outside the range of valid CCSID values.

2 The CCSID specified on the FROMCCSID option and the CCSID of the container are an unsupported combination. The CCSID of the container is the value that was specified, or defaulted, on the first PUT CONTAINER command for this container. The first time each invalid combination is used, CICS issues error message DFHAP0802, which contains the pair of CCSIDs.

4 One or more characters could not be converted. Each unconverted character has been replaced by a blank in the converted data.

5 There was an internal error in the code page conversion of a container. This error can occur only when the target of the PUT is an existing, CICS-created container.

122 CHANNELERR
RESP2 values:

1 The name specified on the CHANNEL option contains an illegal character or combination of characters.

3 Either the current channel or the channel specified on the CHANNEL option is read-only.

125 CODEPAGEERR

RESP2 values:

1 The code page specified on the FROMCODEPAGE option is not supported.

2 The code page specified on the FROMCODEPAGE option and the CCSID of the container are an unsupported combination. The CCSID of the container is the value that was specified using either FROMCODEPAGE or FROMCCSID, or defaulted, on the first PUT CONTAINER command for this container. The first time each invalid combination is used, CICS issues error message DFHAP0802, which contains the pair of CCSIDs.

4 One or more characters could not be converted. Each unconverted character has been replaced by a blank in the converted data. This error can occur only when the target of the PUT is an existing container.

5 There was an internal error in the code page conversion of a container. This error can occur only when the target of the PUT is an existing, CICS-created container.

110 CONTAINERERR

RESP2 values:

18 The name specified on the CONTAINER option contains an illegal character or combination of characters.

Note: This occurrence might happen if the CHANNEL attribute is not set, and that there is no default CHANNEL. The command might be interpreted as PUT CONTAINER (BTS), and be validated according to the BTS naming rules.

16 INVREQ

RESP2 values:

1 The DATATYPE option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one.) DATATYPE is valid only on PUT CONTAINER commands that specify (explicitly or implicitly) a channel.

2 The FROMCCSID option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one.) FROMCCSID is valid only on PUT CONTAINER commands that specify (explicitly or implicitly) a channel.

4 The CHANNEL option was not specified, there is no current channel (because the program that issued the command was not passed one), and the command was issued outside the scope of a currently-active BTS activity.

30 You tried to write to a CICS-defined read only container.

32 A CVDA value other than CHAR or BIT was specified for DATATYPE.

33 An attempt was made to change the data-type of an existing container.

34 A data-type of BIT is invalid with a CCSID.

22 LENGERR
RESP2 values:

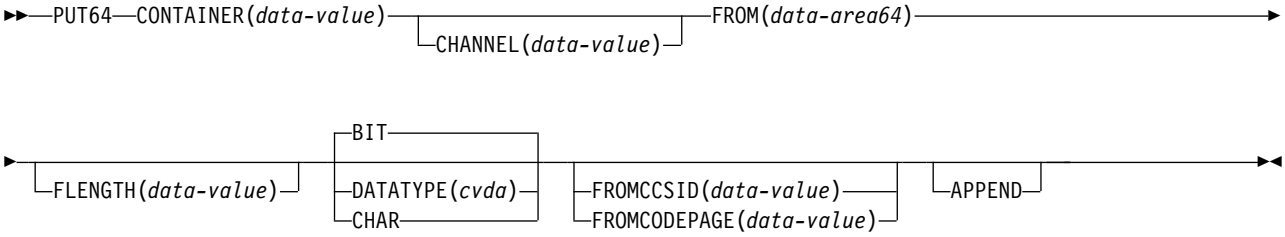
1 A negative number was specified on the FLENGTH option.

PUT64 CONTAINER

Place data from 64-bit storage in a named channel container. This command is for use only in non-Language Environment (LE) AMODE(64) assembler language application programs. CICS business transaction services (BTS) containers are not supported.

See Assembler language programming restrictions and requirements.

PUT64 CONTAINER



Conditions: CCSIDERR, CHANNELERR, CODEPAGEERR, CONTAINERERR, INVREQ, LENGERR

This command is threadsafe.

Description

PUT64 CONTAINER places data from 64-bit storage in a container that is associated with a specified channel.

The container is identified by name. The channel that owns the container can be identified explicitly, by specifying the CHANNEL option, or implicitly, by omitting the CHANNEL option. When this option is omitted, the current channel is implied.

If the named container does not exist, it is created. If the named container exists, its previous contents are overwritten, unless you specify the APPEND option. If the named channel does not exist, it is created.

There is no limit to the number of containers that can be associated with a channel. The size of individual containers is limited only by the amount of storage available.

Note: Be aware that if you create multiple large containers you might limit the amount of storage available to other applications.

Options

APPEND
Specifies that the data passed to the container is appended to the existing data in the container. If this option is not stated, the existing data in the container is overwritten by the data that is passed to the container.

CHANNEL(*data-value*)
Specifies the name (1–16 characters) of the channel that owns the container. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Channel names are always in EBCDIC. The set of allowed characters for channel names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be shipped between regions, it is advisable to restrict the characters that are used to name them to A-Z a-z 0-9 & : = , ; < > . - and _.

If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

You can specify the channel name DFHTRANSACTION to use a transaction channel. A transaction channel does not go out of scope when the link level changes; it is always accessible in the transaction. For more information, see Channels and containers.

CONTAINER(*data-value*)
Specifies the name (1–16 characters) of the container into which data is placed.

The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is fewer than 16 characters, it is padded with trailing blanks up to 16 characters.

Do not use container names that begin with DFH, unless requested to do so by CICS.

Container names are always in EBCDIC. The set of allowed characters for container names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if containers are to be shipped between regions, it is advisable to restrict the characters that are used to name them to A-Z 0-9 & : = , ; < > . - and _.

DATATYPE(*cvda*)
Specifies the type of data to put into the container. This option applies only to new containers. If the container exists, its data type was established when it was created and cannot be changed. CVDA values are as follows:

BIT Bit data. The data in the container cannot be converted. This is the default value, unless FROMCCSID is specified.

CHAR
Character data. The data to store in the container is converted (if required) according to the setting in the FROMCCSID or FROMCODEPAGE value. If the FROMCCSID and FROMCODEPAGE options are not specified, it is assumed that the data is encoded in the CCSID of the region, as specified in the **LOCALCCSID** system initialization parameter.

All the data in a container is converted as if it were a single character string. For SBCS code pages, a structure that consists of several character fields is equivalent to a single-byte character string. However, for DBCS code pages this is not the case. If you use DBCS code pages, you must put each character string into a separate container to ensure that data conversion works correctly.

FLENGTH(*data-value*)
Specifies, as a fullword binary value, the length of the data area from which data is read.

FROM(*data-area64*)
Specifies a 64-bit data area reference to an area from which the data is written to the container. *data-area64* refers to an area that is referenced by a 64-bit pointer and that can be in 64-bit (above-the-bar) storage.

FROMCCSID(*data-value*)
Specifies the current Coded Character Set Identifier (CCSID) of the character data to put into the container, as a fullword binary number. If you prefer to specify an IANA name for the code page, or if you prefer to specify the CCSID as alphanumeric characters, use the FROMCODEPAGE option instead. Use this option if the data to place in the container is not encoded in the CCSID of the region, as specified in the LOCALCCSID system initialization parameter.

If the FROMCCSID option is specified, DATATYPE(DFHVALUE(CHAR)) is implied.

FROMCODEPAGE(*data-value*)
Specifies an IANA-registered alphanumeric charset name or a Coded Character Set Identifier (CCSID) for the current code page of the character data to put into the container, using up to 40 alphanumeric characters, including appropriate punctuation. Use this option instead of the CCSID option if you prefer to use an IANA-registered charset name, as specified in the Content-Type header for an HTTP request. CICS converts the IANA name into a CCSID, and the subsequent data conversion process is identical. Also use this option if you prefer to specify the CCSID in alphanumeric characters, rather than as a fullword binary number.

If the FROMCCSID option is specified, DATATYPE(DFHVALUE(CHAR)) is implied.

Conditions

123 CCSIDERR
RESP2 values:

1 The CCSID specified on the FROMCCSID option is outside the range of valid CCSID values.

2 The CCSID specified on the FROMCCSID option and the CCSID of the container are an unsupported combination. The CCSID of the container is the value that was specified, or defaulted, on the first PUT CONTAINER command for this container. The first time each invalid combination is used, CICS issues error message DFHAP0802, which contains the pair of CCSIDs.

4 One or more characters could not be converted. Each unconverted character is replaced by a blank in the converted data.

5 There was an internal error in the code page conversion of a container. This error can occur only when the target of the PUT is an existing, CICS-created container.

122 CHANNELERR
RESP2 values:

1 The name specified on the CHANNEL option contains an invalid character or combination of characters.

3 Either the current channel or the channel specified on the CHANNEL option is read-only.

125 CODEPAGEERR
RESP2 values:

1 The code page specified on the FROMCODEPAGE option is not supported.

2 The code page specified on the FROMCODEPAGE option and the CCSID of the container are an unsupported combination. The CCSID of the container is the value that was specified using either FROMCODEPAGE or FROMCCSID, or defaulted, on the first PUT CONTAINER command for this container. The first time each invalid combination is used, CICS issues error message DFHAP0802, which contains the pair of CCSIDs.

4 One or more characters could not be converted. Each unconverted character is replaced by a blank in the converted data. This error can occur only when the target of the PUT is an existing container.

5 There was an internal error in the code page conversion of a container. This error can occur only when the target of the PUT is an existing, CICS-created container.

110 CONTAINERERR
RESP2 values:

18 The name specified on the CONTAINER option contains an invalid character or combination of characters.

16 INVREQ
RESP2 values:

1 The DATATYPE option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one.) DATATYPE is valid only on PUT CONTAINER commands that specify (explicitly or implicitly) a channel.

2 The FROMCCSID option was specified without the CHANNEL option, and there is no current channel (because the program that issued the command was not passed one.) FROMCCSID is valid only on PUT CONTAINER commands that specify (explicitly or implicitly) a channel.

4 The CHANNEL option was not specified and there is no current channel (because the program that issued the command was not passed one).

30 You tried to write to a CICS-defined read only container.

32 A CVDA value other than CHAR or BIT was specified for DATATYPE.

33 An attempt was made to change the data-type of an existing container.

34 A data-type of BIT is invalid with a CCSID.

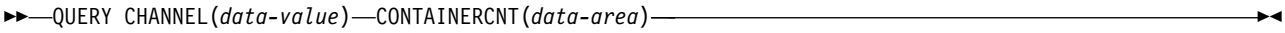
22 LENGERR
RESP2 values:

1 A negative number was specified on the FLENGTH option.

QUERY CHANNEL

Count the number of containers that are in a channel.

QUERY CHANNEL



Conditions: CHANNELERR

This command is threadsafe.

Description

QUERY CHANNEL counts the number of containers that are in a specified channel.

You must specify the CHANNEL option and identify the channel explicitly. You can use the QUERY CHANNEL command with any channel, including the transaction channel (DFHTRANSACTION), and channels that CICS created. To find the name of a program's current channel, that is, the channel with which the program was invoked, use the EXEC CICS ASSIGN CHANNEL command.

Options

CHANNEL(data-value)
Specifies the 1–16 character name of the channel.

CONTAINERCNT(data-area)
Returns, as a fullword binary value, a count of the number of containers that are in the specified channel.

Conditions

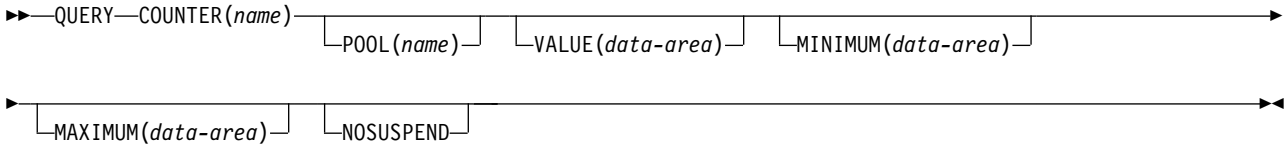
122 CHANNELERR
RESP2 values:

2 The channel specified on the CHANNEL option could not be found.

QUERY COUNTER and QUERY DCOUNTER

Query a named counter. Use COUNTER for fullword signed counters and DCOUNTER for doubleword unsigned counters.

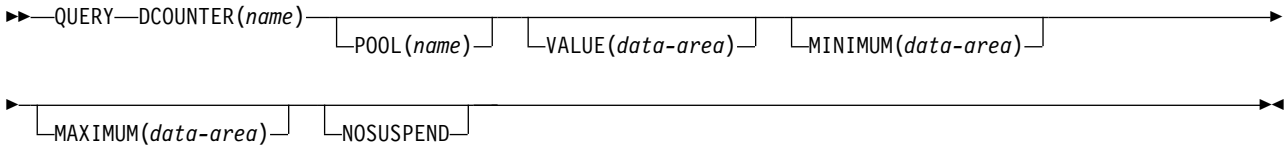
QUERY COUNTER



Conditions: BUSY, INVREQ, LENGERR

This command is threadsafe.

QUERY DCOUNTER



Conditions: BUSY, INVREQ

This command is threadsafe.

Description

These counter commands return the current, maximum, and minimum values for the named counter.

For information about specifying fullword and doubleword variables on these named counter commands, see "CICS command argument values" on page 3.

Options

COUNTER(*name***)**
Specifies the 16-character name of the fullword counter being queried. Valid characters for names are A through Z, 0 through 9, $ @ # and _ (underscore). If *name* is a variable that contains a name that is less than 16 characters, the name must be padded with trailing blanks.

DCOUNTER(*name***)**
Specifies the 16-character name of the doubleword counter being queried. Valid characters for names are A through Z, 0 through 9, $ @ # and _ (underscore). If *name* is a variable that contains a name that is less than 16 characters, the name must be padded with trailing blanks.

MAXIMUM(*data-area***)**
Specifies the data area in which CICS is to return the maximum number for the named counter. CICS returns a fullword signed binary value for the COUNTER command and a doubleword unsigned binary value for the DCOUNTER command.

MINIMUM(*data-area***)**
Specifies the data area in which CICS is to return the minimum number for the

named counter. CICS returns a fullword signed binary value for the COUNTER command and a doubleword unsigned binary value for the DCOUNTER command.

NOSUSPEND
Specifies an immediate return to the application program during a coupling facility structure rebuild. If you omit the NOSUSPEND parameter, the request is retried until it is successful. If you specify NOSUSPEND, the request fails immediately, and CICS returns BUSY, RESP2=500.

POOL(*poolname***)**
Specifies the name of the pool in which the named counter resides.

Valid characters for the pool selector string are A through Z, 0 through 9, $ @ # and _ (underscore). If *name* is a variable that contains a name that is less than 8 characters, the name must be padded with trailing blanks.

This parameter is optional. If you omit the name of the pool, a pool selector value of 8 blanks is assumed.

If there is no matching entry in the DFHNCOPT options table, CICS uses the name specified on the NCPLDFT system initialization parameter, which Specifies the default named counter pool.

For information about generating a named counter options table using the DFHNCO macro, see .

VALUE(*data-area***)**
Specifies the data area in which CICS is to return the current value for the named counter. CICS returns a fullword signed binary value for the COUNTER command and a doubleword unsigned binary value for the DCOUNTER command.

Note that, if the named counter is in the counter-at-limit condition, CICS does not return an exception condition. In this case, CICS returns a normal response with a value that is 1 greater than the maximum value specified or assumed for the counter, using unsigned addition. If the maximum value is the largest positive number that can be held in a signed fullword, the value returned by QUERY COUNTER for a counter-at-limit condition is the largest negative number.

Conditions

128 BUSY
RESP2 values:

500 The NOSUSPEND option was specified on the command, and the coupling facility structure is not currently available during a rebuild.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

201 Named counter not found.

301 The server has reported an error code that is not understood by the named counter interface. Generally, this is not possible unless the interface load module, DFHNCIF, is at a lower maintenance or release level than the server itself.

303 An unexpected error, such as structure failure or loss of connectivity,

has occurred on a macro used to access the coupling facility. Further information is in message DFHNC0441 in the application job log.

304 The pool selection parameter specified in the program cannot be resolved to a valid server name using the current options table.

305 The interface cannot establish a connection to the server for the selected named counter pool. Further information is in an AXM services message (AXMSC*nnnn*) in the application job log.

306 An abend occurred during server processing of a request. Further information is in a message in the application job log and the server job log.

308 The DFHNCOPT options table module, required to resolve a pool name, cannot be loaded.

309 During processing of the options table, the named counter interface encountered an unknown entry format. Either the options table is not correctly generated, or the DFHNCIF interface load module is not at the same release level as the options table.

310 An options table entry that matches the given pool name specified a user exit program, but the user exit program is not link-edited with the options table and cannot be loaded.

311 A response from the named counter server to the client region interface module, DFHNCIF, indicates that a system-managed rebuild is in progress but the EXEC CICS interface does not recognize the condition. This means that the CICS region is at CICS TS 2.1 or earlier.

403 The POOL parameter contains invalid characters or embedded spaces.

404 The COUNTER parameter contains invalid characters or embedded spaces.

Default action: terminate the task abnormally.

22 LENGERR

LENGERR occurs for COUNTER commands only and does not apply to DCOUNTER requests. It occurs when a counter that was defined by a DCOUNTER command or by the CALL interface has a value which is too large to be correctly represented as a fullword signed binary value (that is, the counter uses more than 31 bits).

In each of the three cases of overflow, the named counter server completes the operation, and returns a warning response to CICS, which CICS returns to your application program as the RESP2 value. The data area contains the low-order 32 bits returned from the named counter server, which could be a negative number.

RESP2 values:

001 The current value that the server has attempted to return in one of the data areas has overflowed into the high-order (sign) bit (that is, the value returned is negative).

Note: LENGERR with RESP2=001 cannot occur for a named counter that is in the counter-at-limit condition. If the counter-at-limit condition has been reached, the value (which could be negative) is returned with a normal response.

002 A value is too large for a fullword data area by only 1 bit. In this case, the overflow value is exactly 1.

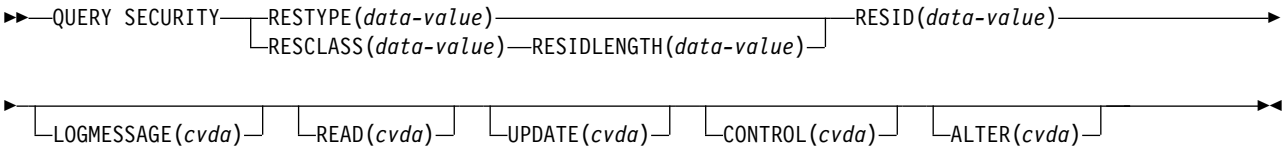
003 A value is too large for a fullword data area by a value greater than 1.

Default action: terminate the task abnormally.

QUERY SECURITY

To query the security authorization of the user.

QUERY SECURITY



Conditions: INVREQ, LENGERR, NOTFND, QIDERR

This command is threadsafe.

Description

The **QUERY SECURITY** command allows the application to determine whether the user has access to resources defined in the external security manager (ESM). These resources can be:

- In CICS resource classes
- In user-defined resource classes

The user in this context is the user invoking the transaction that contains the **QUERY SECURITY** command.

For more information on the use of the **QUERY SECURITY** command, see Security checking using the Query Security command.

Options

ALTER(*cvda***)**
Query whether the user has ALTER authority for the named resource. The cvda values returned by CICS are ALTERABLE and NOTALTERABLE.

CONTROL(*cvda***)**
Query whether the user has CONTROL authority for the named resource. The cvda values returned by CICS are CTRLABLE and NOTCTRLABLE.

LOGMESSAGE(*cvda***)**
Inhibit security violation messages. The values passed to CICS are LOG (the default value), or, to inhibit messages, NOLOG.

READ(*cvda***)**
Query whether the user has READ authority command for the named resource. The cvda values returned by CICS are READABLE and NOTREADABLE. READ access authority usually permits nondestructive use of a resource as, for example, in the case of READ and INQUIRE commands.

RESCLASS(*data-value***)**
Specifies an 8-character field identifying the name of a valid resource class, that could be non-CICS, in the ESM. The class name identified by RESCLASS is treated literally with no translation.

If the ESM is RACF, the class can be CICS-supplied or user-defined. RESCLASS enables you to define more narrowly the authorization to be queried; for example, you can query at the record or field level.

The responses returned by the command reflect the definition of the RESID resource as defined in the specified RESCLASS.

RESID(*data-value***)**
Specifies the name of the CICS or user-defined resource that you want to query the users access to. The value is a character string (1-12 characters for a CICS resource, and 1-246 characters for a user-defined resource, unless you are using the COBOL3 translator option in which case the maximum length is 160 characters).

Note: RESID refers to a CICS-defined resource only when RESTYPE('SPCOMMAND') is specified, otherwise it refers to a user-defined resource. For a list of the CICS RESID values that you can use when RESTYPE('SPCOMMAND') is specified, see RESID values.

Note that the actual resource checked depends on whether RESCLASS or RESTYPE is specified in the command and whether prefixing is active (SECPRFX=YES or SECPRFX=*prefix* specified as a system initialization parameter).

If RESCLASS is specified, the resource checked is always the actual RESID data-value, whether or not prefixing is on or off. IF RESTYPE is specified and SECPRFX=NO, the resource checked is the RESID data-value as specified. Otherwise the resource checked is the RESID data-value prefixed with either the CICS region userid (if SECPRFX=YES), or another prefix (if SECPRFX=*prefix*).

RESIDLENGTH(*data-value***)**
Specifies the length, as a fullword binary, of the resource identifier in RESID. You only use this parameter when specifying the RESCLASS option.

RESTYPE(*data-value***)**
Specifies the type of resource (1–12 characters) you want to query the user's access to.

The responses returned by the command reflect the results that would be obtained if an actual attempt was made to access the specified CICS resource. The value you specify for RESTYPE must be one of the following resource types:

Table 17. QUERY SECURITY RESTYPE values

RESTYPE value	**X***name* **parameter**
ATOMSERVICE	XRES
BUNDLE	XRES
DB2ENTRY	XDB2
DOCTEMPLATE	XRES
EPADAPTER	XRES
EPADAPTERSET	XRES

Table 17. QUERY SECURITY RESTYPE values (continued)

RESTYPE value	**X*name* parameter**
EVENTBINDING	XRES
FILE	XFCT
JOURNALNAME	XJCT
JOURNALNUM 1	XJCT
JVMSERVER	XRES
PROGRAM	XPPT
PSB	XPSB
SPCOMMAND 2	XCMD
TDQUEUE	XDCT
TRANSACTION	XPCT
TRANSATTACH	XTRAN
TSQUEUE	XTST
TSQNAME	XTST
XMLTRANSFORM	XRES

1. Supported for compatibility with earlier releases.
2. SPCOMMAND is a resource type that you can use to specify a RESID for a command.

The **XHFS** system initialization parameter controls resource security for zFS files and does not have a corresponding RESTYPE value on the **QUERY SECURITY** command. Access controls for zFS files follow the system of permissions used by z/OS UNIX System Services, so they operate in a different way.

With dynamic transaction routing, you do not have to install transaction definitions in terminal owning regions. A **QUERY SECURITY** command with a RESTYPE of TRANSATTACH returns the NOTFND condition if the transaction is not installed. Application developers must be aware that the transaction might be routed dynamically.

UPDATE(*cvda***)**
Query whether the user has UPDATE authority for the named resource. The CVDA values returned by CICS are UPDATABLE and NOTUPDATABLE. UPDATE access authority usually permits destructive use of a resource as, for example, in the case of WRITE, DELETE, or UPDATE commands.

Conditions

16 INVREQ
RESP2 values:

7 The cvda value is not valid for the LOGMESSAGE.

9 The RESID is invalid or filled with blanks.

10 The external security manager (ESM) is inactive or not present.

Default action: terminate the task abnormally.

22 LENGERR
RESP2 values:

6 The RESIDLENGTH value is not valid, that is, not in the range 1 through 246.

Default action: terminate the task abnormally.

13 NOTFND

RESP2 values:

1 The RESID is not valid.

2 The RESTYPE is not valid.

3 The RESID value for RESTYPE (SPCOMMAND) is not valid.

5 The RESCLASS is not defined to the external security manager (ESM).

8 The resource is not protected. This is only returned when QUERY SECURITY is used with the RESCLASS option (and never occurs with RESTYPE).

Possible causes include:

- RESCLASS not active.
- No profile found.
- ESM not active.

Default action: terminate the task abnormally.

44 QIDERR

RESP2 values:

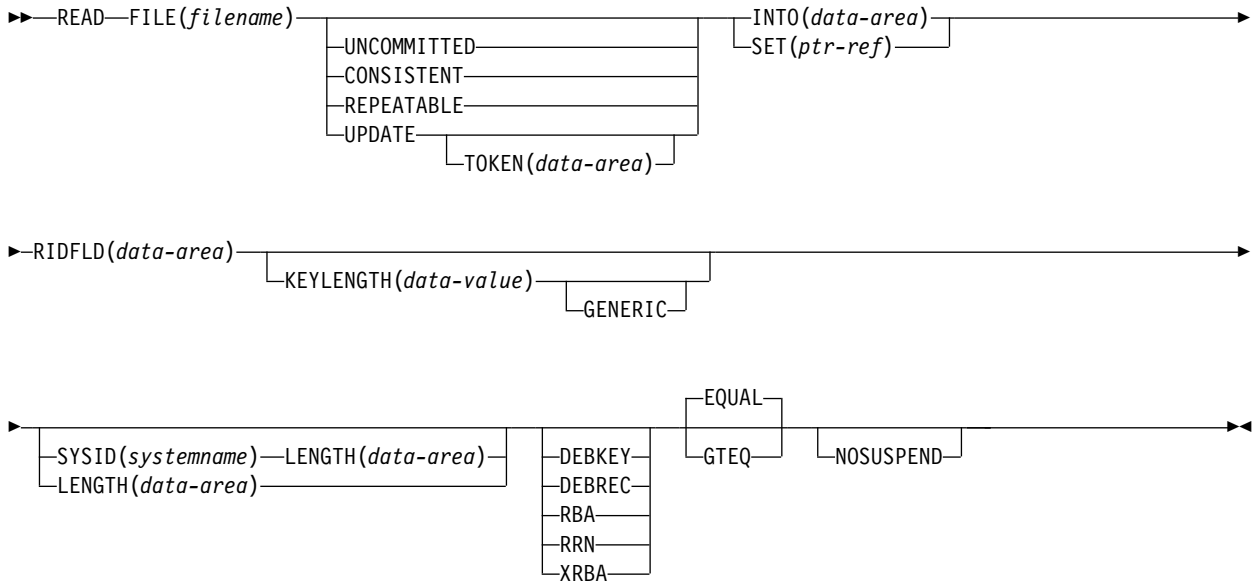
1 An indirect queue name associated with the given RESID is not found.

Default action: terminate the task abnormally.

READ

Read a record from a file.

READ



Conditions: DISABLED, DUPKEY, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, LENGERR, LOADING, LOCKED, NOTAUTH, NOTFND, NOTOPEN, RECORDBUSY, SYSIDERR

This command is threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over an IPIC connection to a remote CICS region.
- Defined as either local VSAM or RLS.

This command is not threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over a non-IPIC connection.
- Defined as a shared data table, coupling facility data table, or BDAM file.

Description

READ reads a record from a file on a local or a remote system.

For both UPDATE and non-UPDATE commands, you must identify the record to be retrieved by the record identification field specified in the RIDFLD option. Immediately upon completion of a READ UPDATE command, the RIDFLD data area is available for reuse by the application program.

Data table considerations

When the READ command reads a CICS-maintained data table, a READ request with UPDATE or RBA is always satisfied by a call to VSAM. A full key read that is neither a generic read nor a READ UPDATE, is satisfied by a reference to the data table if possible. If the record is not found in the table, the source data set is accessed, unless the table is known to be complete, that is, all records in the source are also present in the table (which is the case if loading is finished and none has been rejected by a user exit).

If you carry out a generic read (using the GENERIC option) on a CICS-maintained data table, and CICS returns a NOTFND condition because the record is not found in the table, CICS clears the INTO() and RIDFLD() areas to ensure that an incorrect record is not returned. This behavior optimizes performance, but it differs from the behavior for a generic read of a VSAM file, when the INTO() and RIDFLD() areas are left unchanged in the event of a NOTFND condition. When you convert a VSAM file to a CICS-maintained data table, ensure that any applications that carry out generic reads of the data take appropriate action if a NOTFND condition is returned and the INTO() and RIDFLD() areas are cleared.

When the READ command reads a user-maintained data table, only the data table is accessed once loading is complete; the VSAM file is not changed in any way.

When the READ command reads a coupling-facility data table, only the data table is accessed, even if the table is initially loaded from a VSAM source data set.

If a file that refers to a user-maintained or coupling facility data table is defined with RLSACCESS(YES), the RLS-specific API options CONSISTENT, NOSUSPEND, and REPEATABLE are not supported.

Reading files accessed in RLS mode

When a file is accessed in RLS mode, non-update read requests can specify one of the read integrity options: UNCOMMITTED, CONSISTENT, or REPEATABLE.

If none of these keywords is specified, CICS uses the value specified on the READINTEG parameter of the FILE resource definition, for which the default is UNCOMMITTED.

If you want to use the level of read integrity specified in the READINTEG keyword of the FILE definition, and then you need to change from using a local file to a remote file, or if you change the location of a remote file, ensure that:

- The remote file-owning region supports the read integrity options.
- The FILE definition in the remote system specifies:
  - RLS mode
  - The correct read integrity values for your application.

READ requests that specify the UPDATE keyword, or a CONSISTENT or REPEATABLE read integrity option (either explicitly or implicitly in the FILE definition), return the LOCKED condition if they reference a record that has a retained lock. The key of a locked record is not returned to the application program. Thus, if an application program specifies GTEQ or GENERIC on the READ request it cannot tell which record key is locked.

If a request specifying read integrity is function-shipped to a member of the CICS family of products that does not support read integrity, the request fails:

- If an ISC link is used, the request receives an ATNI abend.
- If an MRO link is used, the request receives an AXF8 abend.

  The AXF8 abend code indicates that your program has attempted to function-ship a request that specifies file control options to a remote CICS region that does not support these options.

Retained and active locks

RECORDBUSY refers to active locks and LOCKED refers to retained locks.

These locks affect READ requests which acquire locks; that is, update requests and requests with read integrity. These are the kinds of READ requests which are referred to in the following bullets. Other READ requests are unaffected by the presence of retained or active locks.

- READ requests for records that have *retained* locks are always rejected with a LOCKED response.
- READ requests for records that have *active* locks wait for the lock to be released, except when the NOSUSPEND keyword is specified, in which case CICS returns the RECORDBUSY response.

Options

CONSISTENT (RLS only)
The record is read with a level of read integrity provided by a VSAM shared lock that lasts for the duration of the request.

If the record is being modified by another task, which therefore holds an exclusive lock, the READ request waits until the update is complete (unless NOSUSPEND is also specified) as follows:

- For a READ request against a non-recoverable file, the READ completes as soon as any VSAM request performing the update completes.
- For a READ request against a recoverable file, the READ request completes when the updating task completes its next sync point or rollback.

DEBKEY
(blocked BDAM) specifies that deblocking is to occur by key. If neither DEBREC nor DEBKEY is specified, deblocking does not occur.

DEBREC
(blocked BDAM) specifies that deblocking is to occur by relative record (relative to zero). If neither DEBREC nor DEBKEY is specified, deblocking does not occur.

EQUAL
specifies that the search is satisfied only by a record having the same key (complete or generic) as that specified in the RIDFLD option.

FILE(*filename*)
specifies the name of the file to be accessed.

If SYSID is specified, the data set to which this file refers is assumed to be on a remote system irrespective of whether the name is defined to CICS. Otherwise, the resource definition is used to find out whether the data set is on a local or a remote system.

GENERIC
(VSAM KSDS, paths and data tables) specifies that the search key is a generic key whose length is specified in the KEYLENGTH option. The search for a record is satisfied when a record is found that has the same starting characters (generic key) as those specified.

GTEQ
(VSAM KSDS, paths and data tables) specifies that, if the search for a record that has the same key (complete or generic) as that specified in the RIDFLD option is unsuccessful, the first record that has a greater key is retrieved.

INTO(*data-area***)**
specifies the data area into which the record retrieved from the data set is to be written.

When INTO is specified, LENGTH must either be specified explicitly or must be capable of being defaulted from the INTO option using the length attribute reference in assembler language, or STG and CSTG in PL/I. LENGTH must be specified explicitly in C.

KEYLENGTH(*data-value***)**
specifies the length (halfword binary) of the key that has been specified in the RIDFLD option, except when RBA or RRN is specified, in which case the KEYLENGTH value is not valid. This option must be specified if GENERIC is specified, and it can be specified whenever a key is specified. However, if the length specified is different from the length defined for the data set and the operation is not generic, the INVREQ condition occurs.

The INVREQ condition also occurs if GENERIC is specified and the KEYLENGTH value is not less than that specified in the VSAM definition.

If KEYLENGTH(0) is used with the object of reading the first record in the data set, the GTEQ option must also be specified. If EQUAL is specified either explicitly or by default with KEYLENGTH(0), the results of the READ are unpredictable.

For remote files, the KEYLENGTH value can be specified in the FILE definition. If KEYLENGTH is not defined there, and is not specified in the application program, and the key is longer than 4 characters, the default value is 4.

LENGTH(*data-area***)**
specifies the length, as a halfword binary value, of the data area where the record is to be put. On completion of the READ command, the LENGTH parameter contains the actual length of the record.

This option must be specified if SYSID is specified.

If the file is on a remote system, the LENGTH parameter need not be set here but must be set in the file resource definition.

If the file is on a local system, the LENGTH parameter must be set for variable-length records, using the INTO option, but not for fixed-length records. It is, however, advisable to specify the length for fixed-length records because:

- It causes a check to be made that the record being read is not too long for the available data area.
- When reading fixed-length records into an area longer than the record being accessed, the LENGERR condition is raised for COBOL, C, PL/I, and assembler-language applications if the LENGTH option is specified. If the LENGTH option is not explicitly specified, then the LENGERR condition is returned if the length can be inferred from the data area. If the length specified exceeds the file record length, CICS uses the longer length for the move. If the target area in the application program is not large enough, storage is overlaid beyond the target area.

If you specify the SET option, you do not need to specify the LENGTH option.

When reading into a target data area that is longer than the record being read, the contents of the target data area, from the end of the retrieved record to the end of the target data area, are unpredictable.

If you specify the INTO option, the LENGTH argument must be a data area that specifies the largest record the program accepts. If the retrieved record is longer than the value specified in the LENGTH option, the record is truncated to the specified value and the LENGERR condition occurs. In this case, the LENGTH data area is set to the length of the record before truncation.

Note that a file control command issued against a variable-length record in a file defined on the local CICS system fails with a LENGERR condition if a length is not specified. However, if the same command is issued against a file defined on a remote system, the command does not fail.

NOSUSPEND (RLS only)
The request does not wait if the record is locked by VSAM with an active lock, including records locked as the result of a DEADLOCK.

Note: Requests that specify NOSUSPEND wait for at least 1 second before CICS returns the RECORDBUSY response.

RBA
(VSAM KSDS or ESDS base data sets, or CICS-maintained data tables only, but not paths) specifies that the record identification field specified in the RIDFLD option contains a relative byte address. This option should be used only when reading records from an ESDS base or when reading from a KSDS base and using relative byte addresses instead of keys to identify the records.

You cannot use RBA for:

- User-maintained data tables
- Coupling facility data tables
- Any KSDS file opened in RLS access mode
- KSDS files that use extended addressing

Also, you are recommended not to use RBA for ESDS files that hold more than 4GB. (Use XRBA instead.)

REPEATABLE (RLS only)
The record is read with a level of read integrity provided by a VSAM shared lock that lasts for the duration of the unit of work in which the read request is issued.

If the record is being modified by another task, which therefore holds an exclusive lock, the READ request waits until the update is complete (unless NOSUSPEND is also specified) as follows:

- For a recoverable file, the READ request completes when the updating transaction completes its next syncpoint or rollback.
- For a non-recoverable file, the READ completes as soon as the VSAM request performing the update completes.

After the READ request has completed, the record remains locked to the task that issued the READ. Other tasks may continue to read the record but no other task is allowed to update the record until the task that issued the READ performs its next syncpoint or rollback.

RIDFLD(*data-area*)
specifies the record identification field. The contents can be a key, a relative byte address, or relative record number (for VSAM data sets), or a block reference, a physical key, and a deblocking argument (for BDAM data sets). For a relative byte address or a relative record number, the format of this field must be fullword binary. For a relative byte address, the RIDFLD value can be

greater than or equal to zero. For a relative record number, the RIDFLD value must be greater than or equal to 1, even when the GTEQ option is specified.

See Identifying VSAM records and for more information about defining the record identification field.

Immediately upon completion of the command, the RIDFLD data area is available for reuse by the application program, even if UPDATE was specified.

Make sure that the variable specified by RIDFLD value is not shorter than the KEYLENGTH specified in this command or, if KEYLENGTH is not specified, the key length of the file you are reading; otherwise, the results are unpredictable.

RRN
(VSAM RRDS) specifies that the record identification field specified in the RIDFLD option contains a relative record number. This option should only be used with files that reference relative record data sets.

SET(*ptr-ref***)**
indicates that CICS is to supply a buffer where the record is read, and specifies the pointer reference that is to contain the address of the retrieved record.

If the DUPKEY condition occurs in assembler language the specified register has not been set. The specified register can be loaded from DFHEITP1.

The pointer reference is valid until the next READ command for the same file or until completion of a corresponding REWRITE, DELETE, or UNLOCK command, or a SYNCPOINT in the case of READ UPDATE SET. If you want to retain the data within the field addressed by the pointer, it should be moved to your own area.

If DATALOCATION(ANY) is associated with the application program, the address of the data can be above or below the 16 MB line.

If DATALOCATION(BELOW) is associated with the application program, the address of the data is below the 16 MB line.

If TASKDATAKEY (USER) is specified for the executing transaction, the data returned is in a user-key; otherwise it is in a CICS-key.

SYSID(*systemname***)**
specifies the name of the system the request is directed to.

If you specify SYSID, and omit both RBA and RRN, you must also specify LENGTH and KEYLENGTH; they cannot be found in the resource definition.

TOKEN(*data-area***)**
specifies, as a fullword binary value, a unique identifier for this READ UPDATE request. This is an output value returned by file control to the requesting task, for use in associating a subsequent REWRITE or DELETE (or UNLOCK) request with the record returned on this READ UPDATE request.

TOKEN can be function shipped. However, if a request specifying TOKEN is function shipped to a member of the CICS family of products that does not recognize this keyword, the request fails.

Note: If you specify TOKEN it implies update.

UNCOMMITTED
The record is read without read integrity.

The current value of the record, as known to VSAM, is returned. No attempt is made to serialize this read request with any concurrent update activity for the

same record. The record may be in the process of being updated by another task, and the record data may change later if that update is subsequently backed out.

UPDATE
specifies that the record is to be obtained for updating or (for VSAM and data tables) deletion. If this option is omitted, a read-only operation is assumed.

UPDATE guarantees read integrity. The mechanism used to ensure data integrity depends on the type of file resource:

- For a VSAM file accessed in RLS mode, the record to be updated is locked by the SMSVSAM server.
- For a VSAM file accessed in non-RLS mode, the record to be updated is locked by CICS and, in addition, the control interval containing the record is held in exclusive control by VSAM.
- For a VSAM file accessed in non-RLS mode, and log(UNDO), CICS holds a record lock until the task syncpoints.
- For a BDAM file, the record to be updated is held in exclusive control by BDAM.
- For a user-maintained data table, the record to be updated is locked by CICS.
- For a CICS-maintained data table, the record to be updated is locked by CICS and, in addition, the control interval containing the record is held in exclusive control by VSAM. The VSAM control interval lock is required because changes to the data table are reflected in the source data set, which is accessed in non-RLS mode.
- For a coupling facility data table using the locking model, the record to be updated is locked by the coupling facility data table server.
- For a coupling facility data table using the contention model, records are not locked, enabling the records to be read for update by more than one task. If a record read for update by one task is then changed by another, the first task is notified, when it issues a REWRITE or DELETE command, by the CHANGED exception condition. If a record read for update by one task is then deleted by another, the first task is notified, when it issues a REWRITE or DELETE command, by the NOTFND condition.

If another task has issued a READ REPEATABLE request against the same record, your READ UPDATE request is made to wait until that task reaches SYNCPOINT (unless you issued NOSUSPEND).

XRBA
specifies that the record identification field specified in the RIDFLD option contains an extended relative byte address. This option should be used when reading records from an ESDS extended addressing data set.

KSDS data sets cannot be accessed by XRBA.

Conditions

84 DISABLED
RESP2 values:

50 A file is disabled because it was initially defined as disabled and has not since been enabled.

A file is disabled because it has been disabled by a SET FILE or a CEMT SET FILE command.

Default action: terminate the task abnormally.

15 DUPKEY

RESP2 values: (VSAM)

140 A record is retrieved by way of an alternate index with the NONUNIQUEKEY attribute, and another alternate index record with the same key follows

In assembler language, if the SET option is being used, the specified register has not been set, but can be loaded from DFHEITP1.

Default action: terminate the task abnormally.

12 FILENOTFOUND

RESP2 values:

1 The file name supplied in the FILE option is not defined to CICS.

Default action: terminate the task abnormally.

21 ILLOGIC

RESP2 values: (VSAM)

110 A VSAM error occurs that is not in one of the other CICS response categories.

See EIBRCODE in the EXEC interface block; for details, see EIB fields.

For user-maintained data tables, this condition occurs only for a non-UPDATE READ during loading when CICS has attempted to retrieve the record from the source data set.

Default action: terminate the task abnormally.

16 INVREQ

RESP2 values:

20 READ is not allowed according to the resource definition.

A READ command with the UPDATE option is issued to a file where update operations are not allowed according to the resource definition.

25 The KEYLENGTH and GENERIC options are specified, and the length specified in the KEYLENGTH option is greater than or equal to the length of a full key.

26 The KEYLENGTH option is specified (but the GENERIC option is not specified), and the specified length does not equal the length defined for the data set to which this file refers.

28 Following a READ UPDATE command without TOKEN, another READ UPDATE without TOKEN was issued against the same file without an intervening REWRITE, DELETE without RIDFLD specified, UNLOCK, or SYNCPOINT command. This condition may in some cases be raised despite the fact that the first READ UPDATE was not successful, for example because it timed out.

40 A BDAM key conversion error occurred.

42 The KEYLENGTH and GENERIC options are specified, and the length specified in the KEYLENGTH option is less than zero.

44 The command does not conform to the format of READ for a user-maintained or coupling facility data table; for example, if RBA is specified.

51 A READ to a KSDS file that is being accessed in RLS mode specifies the RBA keyword. RLS mode does not support relative byte address access to KSDS data sets.

52 CONSISTENT is specified on a READ request to a non-RLS mode file, or to a data table that is specified with RLSACCESS(YES). CONSISTENT is not allowed if the file refers to a data table, even if the file definition specifies RLSACCESS(YES).

53 REPEATABLE is specified on a READ request to a non-RLS mode file or to a data table that is specified with RLSACCESS(YES). REPEATABLE is not allowed if the file refers to a data table, even if the file definition specifies RLSACCESS(YES).

55 NOSUSPEND is specified on a READ request to a non-RLS mode file or to a data table that is specified with RLSACCESS(YES). NOSUSPEND is not allowed if the file refers to a data table, even if the file definition specifies RLSACCESS(YES).

56 An attempt to update a recoverable coupling facility data table has failed because the current unit of work has already updated 1024 recoverable coupling facility data tables. You cannot update more than 1024 recoverable coupling facility data tables within a unit of work.

57 An attempt to update the file has failed because the associated VSAM data set has an AVAILABILITY state of RREPL, which disallows update operations except those from replication programs.

59 XRBA was specified, but the data set is not an extended ESDS.

Default action: terminate the task abnormally.

17 IOERR

RESP2 values:

120 There is an I/O error during the READ operation. An I/O error is any unusual event that is not covered by a CICS condition.

For VSAM files, IOERR normally indicates a hardware error.

For user-maintained data tables, this condition occurs only for a non-UPDATE READ during loading when CICS has attempted to retrieve the record from the source data set.

For a coupling facility data table, an IOERR indicates a bad response returned from a coupling facility access.

Further information is available in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

54 ISCINVREQ

RESP2 values:

70 The remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

22 LENGERR

RESP2 values:

10 Neither the LENGTH option nor the SET option is specified on a

READ command for a file with variable-length records or for a BDAM file with variable-length or undefined-format records.

11 The length of a record read with the INTO option specified exceeds the value specified in the LENGTH option; the record is truncated, and the data area supplied in the LENGTH option is set to the actual length of the record.

13 An incorrect length is specified for a file with fixed-length records.

Default action: terminate the task abnormally.

94 LOADING

RESP2 values:

104 The request cannot be satisfied because it is issued against a data table that is still being loaded. The condition can be raised for one of the following reasons:

- The READ specifies a record that has not yet been loaded into a coupling facility data table. Records can be read or modified while a CFDT is loading only if the requested key is within the range of those records already loaded.

  The LOADING response can also be returned for a coupling facility data table that has failed during loading. For more information about what happens if the load for a coupling facility data table fails, see the description of the XDTLC global user exit in Data tables management exits (XDTRD, XDTAD, and XDTLC) .
- The READ specifies the UPDATE option for a user-maintained data table. A user-maintained data table cannot be modified during loading.
- The READ specifies the GENERIC or GTEQ options for a user-maintained data table. While a UMT is being loaded, you can use read requests with precise keys only.

If your application programs encounter the LOADING condition persistently or too frequently, check that this is not caused by conflicting file definitions that reference the same data set.

Default action: terminate the task abnormally.

100 LOCKED

RESP2 values:

106 An attempt is being made to read a record either specifying the UPDATE keyword, or specifying (explicitly or implicitly) CONSISTENT or REPEATABLE, but the record is locked by a retained lock (see "Retained and active locks" on page 375).

The LOCKED condition can also occur for a READ UPDATE request to a recoverable CFDT that uses the locking model, if the record being read is locked by a retained lock. See the Coupling facility data table retained locks for information about investigating retained locks on records in a coupling facility data table.

Default action: abend the task with code AEX8.

70 NOTAUTH

RESP2 values:

101 A resource security check has failed on FILE(filename).

Default action: terminate the task abnormally.

13 NOTFND

RESP2 values:

80 An attempt to retrieve a record based on the search argument provided is unsuccessful. For data tables, this condition occurs if an attempt to read a record is unsuccessful because there is no entry with the specified key in the data table. This does not mean that there is no such record in the source data set (if the table was created from one); it may be that such a record is present but was either rejected during initial loading by the user exit XDTRD, or was subsequently deleted from the data table. For remote files, this condition occurs if an attempt to read a record is made without keylength specified either in the application or the file definition, and the actual key is longer than 4 characters.

81 XRBA was specified, and the value of RIDFLD was greater than 4 GB, but the data set is not an extended ESDS.

Default action: terminate the task abnormally.

19 NOTOPEN

RESP2 values:

60 NOTOPEN (RESP2 60) is returned for one of the following reasons:

- The requested file is CLOSED and UNENABLED. The CLOSED, UNENABLED state is reached after a CLOSE request has been received against an OPEN ENABLED file and the file is no longer in use. You can also make CLOSED, UNENABLED the initial state, by specifying STATUS(UNENABLED) and OPENTIME(FIRSTREF) on the FILE resource definition. (For BDAM files, you use the FILSTAT parameter of DFHFCT TYPE=FILE.)
- The requested file is OPEN and in use by other transactions, but a CLOSE request against the file has been received.
- A READ command is issued against a data set that is quiesced, or is being quiesced, as a result of a SET DSNAME QUIESCED or IMMQUIESCED command.
- The requested file is CLOSED and ENABLED, so CICS has tried to open the file as part of executing the request. This file open has failed for some reason. You should examine the console for messages that explain why the file open has been unsuccessful.

This condition does not occur if the request is made to a CLOSED, DISABLED file. In this case, the DISABLED condition occurs.

Default action: terminate the task abnormally.

101 RECORDBUSY

RESP2 values:

107 The NOSUSPEND keyword is specified and the record is locked by an active lock (see "Retained and active locks" on page 375).

Default action: abend the task with code AEX9.

53 SYSIDERR

RESP2 values:

130 The SYSID option specifies a name that is neither the local system nor a remote system that is defined by a CONNECTION or IPCONN definition. SYSIDERR also occurs when the link to the remote system is known but unavailable. In the case of an IPCONN, SYSIDERR occurs if

the link is known but either the local or remote systems do not support file control commands that are function shipped using IP interconnectivity.

131 For a coupling facility data table, the connection to the coupling facility data table server has failed. This could be because the server itself has failed, or the server is available, but CICS has failed to connect to it.

132 The READ is issued for a coupling facility data table that no longer exists, probably because of a coupling facility failure, in which case the coupling facility data table server also fails. See the Setting up and running a coupling facility data table server for information about restarting a coupling facility data table server and reloading a table.

Default action: terminate the task abnormally.

Examples

The following example shows you how to read a record from a file named MASTER into a specified data area:

```
EXEC CICS READ
     INTO(RECORD)
     FILE('MASTER')
     RIDFLD(ACCTNO)
```

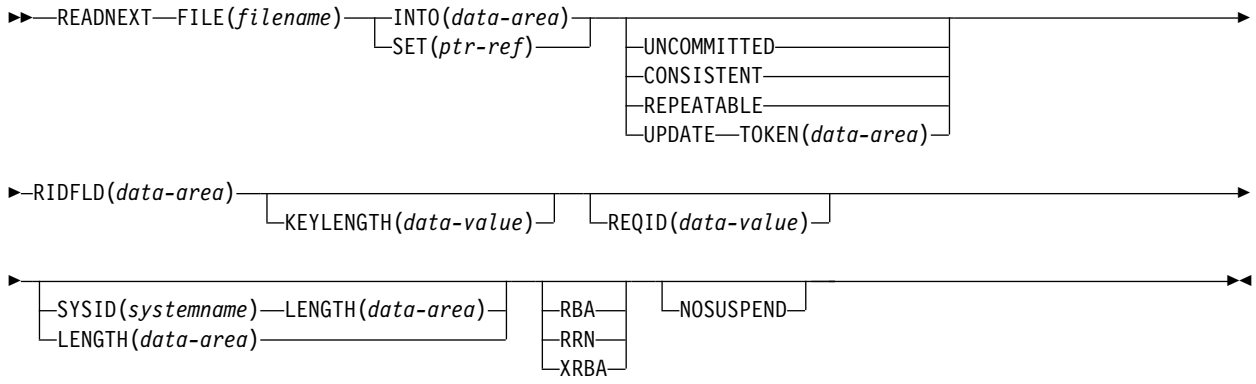
The following example shows you how to read a record for update from a VSAM file using a generic key and specifying a greater-or-equal key search.

```
EXEC CICS READ
     INTO(RECORD)
     LENGTH(RECLEN)
     FILE('MASTVSAM')
     RIDFLD(ACCTNO)
     KEYLENGTH(4)
     GENERIC
     GTEQ
     UPDATE
```

READNEXT

Read next record during a browse of a file.

READNEXT



Conditions: DUPKEY, ENDFILE, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, LENGERR, LOADING, LOCKED, NOTAUTH, NOTFND, RECORDBUSY, SYSIDERR

This command is threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over an IPIC connection to a remote CICS region.
- Defined as either local VSAM or RLS.

This command is not threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over a non-IPIC connection.
- Defined as a shared data table, coupling facility data table, or BDAM file.

Description

READNEXT can be used repeatedly to read records in sequential order from a file on a local or a remote system. Such a series of sequential read commands is known as a *browse* of the file. A browse can also consist of a sequence of READNEXT and READPREV commands in any order. A browse must be initiated with the STARTBR command, to identify the starting point of the browse, and terminated with the ENDBR command.

You must provide, in the RIDFLD option, a data area that is sufficiently large to contain a complete identifier (full key, RBA, or RRN) of records in the file. This data area can be used both as an output and as an input parameter.

It is used as an output parameter when CICS, on completion of each READNEXT command, places the complete identifier of the record just retrieved into the RIDFLD data area. CICS then holds this identifier to mark the point from which the subsequent READNEXT is to continue.

It may, except for BDAM, also be used as an input parameter. Modifying the RIDFLD before issuing the next READNEXT command causes that command to reposition the browse to the new identifier, from which it continues in the usual way. If the browse was started with the GENERIC option, the modified RIDFLD must be generic. If the browse was started with the GTEQ option, the next record returned is the first record in the data set with a key greater than or equal to the modified RIDFLD.

A READNEXT command following a READPREV, or a STARTBR or RESETBR that specified a 'last' key value, is treated as though the RIDFLD value has been modified, and results in a reposition (as above).

Reading files accessed in RLS mode

For files accessed in RLS mode, you can include the UPDATE keyword on the READNEXT request to update some records during the browse. If you specify UPDATE you must also specify TOKEN. You can then update the record by issuing a DELETE or REWRITE command that specifies the TOKEN returned on the browse function.

Note: TOKEN without the UPDATE keyword implies UPDATE.

Use of the UPDATE option is subject to the following rules:

- You can specify UPDATE on a READNEXT command only if the file is accessed in RLS mode. If you specify UPDATE for a file accessed in non-RLS mode, CICS returns INVREQ.
- You can specify UPDATE on READNEXT, but not on the STARTBR or RESETBR commands.
- You can intermix UPDATE and non-update requests within the same browse.
- CICS does not preserve the UPDATE option for you from one READNEXT command to the next.

  CICS supports only one TOKEN in a browse sequence, and the TOKEN value on each READNEXT invalidates the previous value.

Locks for UPDATE

Specifying UPDATE on a READNEXT command acquires an exclusive lock. The duration of these exclusive locks within a browse depends on the action your application program takes:

- If you decide to DELETE or REWRITE the last record acquired by a READNEXT UPDATE in a browse, using the associated token, the lock remains active as follows:
  - If the file is recoverable, the lock is released at completion of the next syncpoint or rollback.
  - If the file is non-recoverable, the lock will be released by the time ENDBR has completed, but might be released earlier.
- If you decide *not* to update the last record read, CICS frees the exclusive lock when your program either issues another READNEXT or READPREV command, or ends the browse.

Important: The UNLOCK command does *not* free an exclusive lock held by VSAM against a record acquired by READNEXT UPDATE. An UNLOCK in a browse invalidates the TOKEN returned by the last request.

Locks for read integrity

Specifying one of the read integrity options acquires a shared lock on each READNEXT. The duration of these shared locks with a browse depends on the type of read integrity you specify:

- If you specify CONSISTENT read integrity, the shared lock is held only for the duration of each read request—that is, until the record is returned to your program.
- If you specify REPEATABLE read integrity, the shared lock is held for the duration of the unit of work in which the browse is performed. In this case, your program could acquire a large number of shared locks, which will prevent the granting of exclusive locks for update functions. You should use REPEATABLE read integrity in a browse with caution.

Function shipping READNEXT with UPDATE or read integrity

If a READNEXT command that specifies UPDATE or one of the read integrity options is function-shipped to a member of the CICS family of products that does not support UPDATE or the read integrity options, the request fails:

- If an ISC link is used, the request receives an ATNI abend.
- If an MRO link is used, the request receives an AXF8 abend.

  AXF8 is an abend code, received by the sending side of a function-shipped request. It indicates that an attempt has been made to send a request that specifies UPDATE on an MRO link to a CICS region that does not support update or read integrity options.

Read integrity

When a file is accessed in RLS mode, non-update read requests can specify read integrity options: UNCOMMITTED, CONSISTENT, or REPEATABLE.

If you do not specify any of these keywords, CICS uses the value specified on the READINTEG parameter of the FILE resource definition, for which the default is UNCOMMITTED.

If you want to use the level of read integrity specified in the READINTEG keyword of the FILE definition, and then you need to change from using a local file to a remote file, or if you change the location of a remote file, ensure that:

- The remote file-owning region is at the CICS Transaction Server for OS/390®, Version 1 Release 1 (or later) level.
- The FILE definition in the remote system specifies:
  - RLS mode
  - The correct read integrity values for your application.

Retained and active locks

RECORDBUSY refers to active locks, and LOCKED refers to retained locks.

These locks affect READNEXT requests which acquire locks; that is, update requests and requests with read integrity. These are the kinds of READNEXT requests which are referred to in the following bullets. Other READNEXT requests are unaffected by the presence of retained or active locks.

- READNEXT requests for records that have *retained* locks are always rejected with a LOCKED response.
- READNEXT requests for records that have *active* locks wait for the lock to be released, except when the NOSUSPEND keyword is specified, in which case CICS returns the RECORDBUSY response.

Options

CONSISTENT (RLS only)
The record is read with a level of read integrity provided by a VSAM shared lock that lasts for the duration of the request.

If the record is being modified by another task, which therefore holds an exclusive lock, the READNEXT request waits until the update is complete (unless NOSUSPEND is also specified) as follows:

- For a READNEXT request against a non-recoverable file, the READ completes as soon as any VSAM request performing the update completes.
- For a READNEXT request against a recoverable file, the READ request completes when the updating task completes its next sync point or rollback.

FILE(*filename*)
specifies the name of the file to be browsed.

If SYSID is specified, the data set to which this file refers is assumed to be on a remote system irrespective of whether the name is defined to CICS. Otherwise, the resource definition is used to find out whether the data set is on a local or a remote system.

INTO(*data-area*)
specifies the data area into which the record retrieved from the data set is to be written.

KEYLENGTH(*data-value*)
specifies the length (halfword binary) of the key that has been specified in the RIDFLD option, except when RBA or RRN is specified, in which case KEYLENGTH is not valid.

If the browse was started without the GENERIC option (that is a full key browse) and if the length specified is different from the length defined for the data set, the INVREQ condition occurs.

If the browse was started with the GENERIC option (that is, a generic key browse), and if the length specified is greater than the length specified for the data set, the INVREQ condition occurs.

If GTEQ and GENERIC were specified on the most recent STARTBR or RESETBR command, issuing READNEXT with KEYLENGTH(0) specified repositions the BROWSE at the start of the file. If EQUAL had been specified, the effect of READNEXT KEYLENGTH(0) would be unpredictable.

For a generic browse, CICS maintains a current key length for the browse. The current key length is initialized to be the value specified as KEYLENGTH on the STARTBR command.

You can modify the current key length by specifying KEYLENGTH on a READNEXT or RESETBR command. If the current key length is changed, this causes the browse to be repositioned. The browse is repositioned to the key whose initial characters match the value specified in the RIDFLD for the current key length.

The current key length is zero after a request that specifies KEYLENGTH(0).

IF KEYLENGTH is omitted on a READNEXT command, the current key length remains the same and the browse continues without repositioning.

If KEYLENGTH is specified on a READNEXT command and is equal to the current key length, this is treated as being no change and the browse is not

repositioned. The one exception to this is when KEYLENGTH(0) is specified, which always causes the browse to be repositioned to the beginning of the file.

KEYLENGTH can be specified during a generic browse with a value equal to the full key length. This does not cause the current key length to change and the browse is not repositioned. The ability to specify the full key length during a generic browse is provided to allow requests that specify SYSID to be able to tell the function-shipping transformers how long the key is, so that the transformers can ship the key to the file-owning region.

A browse can be repositioned by modifying the RIDFLD data area. A generic browse is repositioned only if the modification to RIDFLD changes the part of RIDFLD corresponding to the current key length. A consequence of this is that the browse cannot be repositioned by modifying the RIDFLD data area if the current key length is zero.

For remote files, the KEYLENGTH can be specified in the FILE definition. If KEYLENGTH is not defined there, and is not specified in the application program, and the key is longer than 4 characters, the default value is 4.

LENGTH(*data-area***)**
specifies the length, as a halfword binary value, of the data area where the record is to be put. On completion of the READNEXT command, the LENGTH parameter contains the actual length of the record.

This option must be specified if SYSID is specified.

If the file is on a remote system, the LENGTH parameter need not be set here but must be set in the file resource definition.

If the file is on a local system, the LENGTH parameter must be set for variable-length records, using the INTO option, but not for fixed-length records. It is, however, advisable to specify the length for fixed-length records because:

- It causes a check to be made that the record being read is not too long for the available data area.
- When browsing fixed-length records into an area longer than the record being accessed, the LENGERR condition is raised for COBOL, C, PL/I, and assembler-language applications if the LENGTH option is specified. If the LENGTH option is not explicitly specified, then the LENGERR condition is returned if the length can be inferred from the data area. If the length specified exceeds the file record length, CICS uses the longer length for the move. If the target area in the application program is not large enough, storage is overlaid beyond the target area.

If you specify the SET option, you need not also specify the LENGTH option.

When browsing into a target data area longer than the record being read, the contents of the target data area, from the end of the retrieved record to the end of the target data area, are unpredictable.

If you specify the INTO option, the LENGTH argument must be a data area that specifies the largest record the program accepts. If the retrieved record is longer than the value specified in the LENGTH option, the record is truncated to the specified value and the LENGERR condition occurs. In this case, the LENGTH data area is set to the length of the record before truncation.

Note that a file control command issued against a variable-length record in a file defined on the local CICS system fails with a LENGERR condition if a length is not specified. However, if the same command is issued against a file defined on a remote system, the command does not fail.

NOSUSPEND (RLS only)
The request does not wait if the record is locked by VSAM with an active lock, including records locked as the result of a DEADLOCK.

Note: Requests that specify NOSUSPEND wait for at least 1 second before CICS returns the RECORDBUSY response.

RBA
(VSAM KSDS or ESDS base data sets, or CICS-maintained data tables only, but not paths) specifies that the record identification field specified in the RIDFLD option contains a relative byte address.

This option must be specified when the STARTBR or RESETBR command specified the RBA option. It must not be specified when the STARTBR or RESETBR command did not specify RBA.

You cannot use RBA for:

- User-maintained data tables
- Coupling facility data tables
- Any KSDS file opened in RLS access mode
- KSDS files that use extended addressing

Also, you are recommended not to use RBA for ESDS files that hold more than 4GB. (Use XRBA instead.)

REPEATABLE (RLS only)
The record is read with a level of read integrity provided by a VSAM shared lock that lasts for the duration of the unit of work in which the read request is issued.

If the record is being modified by another task, which therefore holds an exclusive lock, the READNEXT request waits until the update is complete (unless NOSUSPEND is also specified) as follows:

- For a recoverable file, the READNEXT request completes when the updating transaction completes its next sync point or rollback.
- For a non-recoverable file, the READNEXT completes as soon as the VSAM request performing the update completes.

After the READNEXT request has completed, the record remains locked to the task that issued the READNEXT. Other tasks may continue to read the record but no other task is allowed to update the record until the task that issued the READNEXT performs its next syncpoint or rollback.

REQID(*data-value***)**
specifies as a halfword binary value a unique request identifier for a browse, used to control multiple browse operations on a file. If this option is not specified, a default value of zero is assumed.

RIDFLD(*data-area***)**
specifies the record identification field. The contents can be a key, a relative byte address, or relative record number (for VSAM data sets), or a block reference, physical key, and deblocking argument (for BDAM data sets). For a relative byte address or a relative record number, the format of this field must be fullword binary. For a relative byte address, the RIDFLD can be greater than or equal to zero. For a relative record number, the RIDFLD value can be greater than or equal to 1.

See and for more information about defining the record identification field.

Even if the browse is generic, this RIDFLD should always be large enough to accommodate the complete record identifier. This is because, on completion of the READNEXT command, the field is updated by CICS with the complete identification of the record retrieved.

RRN
(VSAM RRDS) specifies that the record identification field specified in the RIDFLD option contains a relative record number.

SET(*ptr-ref***)**
specifies the pointer reference that is to be set to the address of the retrieved record.

In assembler language, if the DUPKEY condition occurs, the register specified will not have been set, but can be loaded from DFHEITP1.

The pointer reference is valid until the next READNEXT or READPREV command specifies SET for the same browse (REQID) for the same file. The pointer is no longer valid after an ENDBR or SYNCPOINT command. If you want to retain the data within the field addressed by the pointer, you must move it to your own area.

If DATALOCATION(ANY) is associated with the application program, the address of the data can be above or below the 16 MB line.

If DATALOCATION(BELOW) is associated with the application program, the address returned in the SET pointer is below the 16 MB line.

If TASKDATAKEY(USER) is specified for the running task, the data returned is in user-key storage; otherwise it is in CICS-key storage.

SYSID(*systemname***)**
specifies the name of the system to which the request is directed.

If you specify SYSID, and omit both RBA and RRN, you must also specify LENGTH and KEYLENGTH; they cannot be found in the resource definition.

TOKEN(data-area) (RLS only)
Returns, as a fullword binary value, the request identifier for this READNEXT UPDATE request. This is an output value returned by file control to the requesting task, for use in associating a subsequent REWRITE or DELETE (or UNLOCK) request with the record returned on this READNEXT command.

You must specify the returned TOKEN on a subsequent REWRITE or DELETE command to identify the record being rewritten or deleted. You can also specify the value returned by CICS on the TOKEN option on a subsequent UNLOCK command, to identify the token that is being invalidated.

You must specify TOKEN if you specify UPDATE (but UPDATE is assumed if you specify TOKEN without UPDATE).

CICS supports only one active TOKEN at a time for a given REQID. Thus, a TOKEN value remains valid only until the following READNEXT, READPREV, RESETBR, or ENDBR command for this browse, or until a REWRITE, DELETE, or UNLOCK command (whichever is first).

TOKEN can be function shipped. However, if a request specifying TOKEN is function shipped to a member of the CICS family of products that does not recognize this keyword, the request fails.

UNCOMMITTED
The record is read without read integrity. The current data, as known to VSAM, is returned. No attempt is made to serialize this read request with any

concurrent update activity for the same record. The record may be being updated by another transaction, therefore the value of the record may change later if that update is subsequently backed out.

UPDATE (RLS only)
Specifies that the record is to be obtained for updating or deletion. If this option and TOKEN are both omitted, read only is assumed.

If you specify UPDATE, you must also specify TOKEN.

UPDATE is only valid for files defined to the local region

XRBA
specifies that the record identification field specified in the RIDFLD option contains an extended relative byte address. This option should be used when browsing records in an ESDS extended addressing data set.

You cannot specify XRBA on a READNEXT command unless the associated STARTBR or RESETBR command also specified XRBA.

KSDS data sets cannot be accessed by XRBA.

Conditions

15 DUPKEY
RESP2 values (VSAM):

140 A record is retrieved by way of an alternate index with the NONUNIQUEKEY attribute, and another alternate index record with the same key follows. It does not occur as a result of a READNEXT command that reads the last of the records having the nonunique key.

In assembler language, if the SET option is being used, the register specified will not have been set, but can be loaded from DFHEITP1.

Default action: terminate the task abnormally.

20 ENDFILE
RESP2 values:

90 An end-of-file condition is detected during the browse.

Default action: terminate the task abnormally.

12 FILENOTFOUND
RESP2 values:

1 A file name referred to in the FILE option is not defined to CICS.

Default action: terminate the task abnormally.

21 ILLOGIC
The browse that is currently in progress is terminated when this condition is raised.

RESP2 values (VSAM):

110 A VSAM error occurs that is not in one of the other CICS response categories.

See EIBRCODE in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

20 The FILE definition does not allow updates.

25 The KEYLENGTH option is specified for a generic browse (that is, one where GENERIC was specified on the STARTBR or the last RESETBR) and the value of KEYLENGTH was greater than the full key length.

26 The KEYLENGTH option is specified for a nongeneric browse, and the specified length does not equal the length defined for the data set to which this file refers.

34 The REQID, SYSID, or file name does not match that of any successful STARTBR command, or a SYNCPOINT was issued causing STARTBR, READNEXT/READPREVs position to be lost.

37 The type of record identification (for example, key or relative byte address) used to access a data set during the browse has been changed. You cannot specify one type of addressing on STARTBR and another on READNEXT.

42 The KEYLENGTH option is specified for a generic browse (that is one where GENERIC was specified on the STARTBR or the last RESETBR) and the value of KEYLENGTH was less than zero.

52 CONSISTENT is specified on a READ request to a non-RLS mode file, or to a data table that is specified with RLSACCESS(YES). CONSISTENT is not allowed if the file refers to a data table, even if the file definition specifies RLSACCESS(YES).

53 REPEATABLE is specified on a READ request to a non-RLS mode file, or to a data table that is specified with RLSACCESS(YES). REPEATABLE is not allowed if the file refers to a data table, even if the file definition specifies RLSACCESS(YES).

54 UPDATE is not allowed because the file is not a VSAM file that is accessed in RLS mode.

55 NOSUSPEND is specified on a READ request to a non-RLS mode file, or to a data table that is specified with RLSACCESS(YES). NOSUSPEND is not allowed if the file refers to a data table, even if the file definition specifies RLSACCESS(YES).

57 An attempt to update the file has failed because the associated VSAM data set has an AVAILABILITY state of RREPL, which disallows update operations except those from replication programs.

Default action: terminate the task abnormally.

17 IOERR

RESP2 values:

120 There is an I/O error during the READNEXT command. An I/O error is any unusual event that is not covered by a CICS condition.

For VSAM files, IOERR usually indicates a hardware error.

For a coupling facility data table, an IOERR indicates a bad response returned from a coupling facility access.

Further information is available in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

54 ISCINVREQ

RESP2 values:

70 The remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

22 LENGERR

RESP2 values:

10 Neither the LENGTH nor the SET option is specified for a file with variable-length records or a BDAM file with undefined-format records.

11 The length of the record read with the INTO option specified exceeds the value specified in the LENGTH option; the record is truncated, and the data area supplied in the LENGTH option is set to the actual length of the record.

13 An incorrect length is specified for a file with fixed-length records.

Default action: terminate the task abnormally.

94 LOADING

RESP2 values:

104 The read request specifies a record key for a record in a coupling facility data table that is still being loaded, and the key is out of range of the records already loaded. Records can be browsed in a coupling facility data table during loading only if the requested key is within the range of those records already loaded.

The LOADING response can also be returned for a coupling facility data table that has failed during loading. For more information about what happens if the load for a coupling facility data table fails, see the description of the XDTLC global user exit in Data tables management exits (XDTRD, XDTAD, and XDTLC).

If your application programs encounter the LOADING condition persistently or too frequently, check that this is not caused by conflicting file definitions that reference the same data set.

100 LOCKED

RESP2 values:

106 The read request specifies the UPDATE keyword, or one of the read integrity keywords CONSISTENT or REPEATABLE, or the file resource definition specifies read integrity, but VSAM holds a retained lock against the record (see "Retained and active locks" on page 387).

The key of the locked record is not returned to your application program. If you handle this condition and control is returned to your program, the browse can continue and retrieve the record following the locked record by issuing another READNEXT request.

The LOCKED condition can also occur for a request to a recoverable CFDT that uses the locking model, if the record that is being read is locked by a retained lock. See Coupling facility data table retained locks for information about investigating retained locks on records in a coupling facility data table.

Default action: abend the task with code AEX8.

70 NOTAUTH

RESP2 values:

101 A resource security check has failed on FILE(filename).

Default action: terminate the task abnormally.

13 NOTFND
RESP2 values:

80 An attempt to retrieve a record based on the search argument provided is unsuccessful. This may occur if the READNEXT command immediately follows a STARTBR command that specified the key of the last record in the data set (a complete key of X'FF's).

81 XRBA was specified, and the value of RIDFLD was greater than 4 GB, but the data set is not an extended addressing ESDS.

Default action: terminate the task abnormally.

101 RECORDBUSY
RESP2 values:

107 NOSUSPEND is specified on the request but VSAM holds an active lock against the record, which would cause the request to wait (see "Retained and active locks" on page 387).

The key of the locked record is not returned to your application program. If you handle this condition and control is returned to your program, the browse can continue and retrieve the record following the locked record by issuing another READNEXT request.

Default action: abend the task with code AEX9.

53 SYSIDERR
RESP2 values:

130 The SYSID option specifies a name that is neither the local system nor a remote system that is defined by a CONNECTION or IPCONN definition. SYSIDERR also occurs when the link to the remote system is known but unavailable. In the case of an IPCONN, SYSIDERR occurs if the link is known but either the local or remote systems do not support file control commands that are function shipped using IP interconnectivity.

131 For a coupling facility data table, the connection to the coupling facility data table server has failed. This could be because the server itself has failed, or the server is available, but CICS has failed to connect to it.

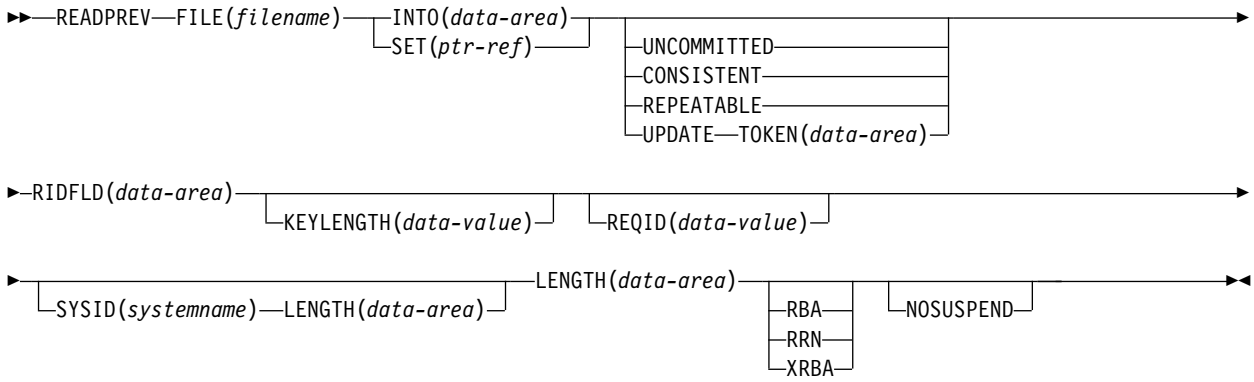
132 The READNEXT is issued against a coupling facility data table that no longer exists, probably because of a coupling facility failure, in which case the coupling facility data table server also fails. See Setting up and running a coupling facility data table server for information about restarting a coupling facility data table server and reloading a table.

Default action: terminate the task abnormally.

READPREV

Read previous record during a file browse; VSAM and data tables only.

READPREV



Conditions: DUPKEY, ENDFILE, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, LENGERR, LOCKED, NOTAUTH, NOTFND, RECORDBUSY, SYSIDERR

This command is threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over an IPIC connection to a remote CICS region.
- Defined as either local VSAM or RLS.

This command is not threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over a non-IPIC connection.
- Defined as a shared data table, coupling facility data table, or BDAM file.

Description

READPREV can be used repeatedly to read records in reverse sequential order from a VSAM file on a local or a remote system.

Such a series of sequential read commands is known as a *browse* of the file. A browse can also consist of a sequence of READNEXT and READPREV commands in any order. A browse must be initiated with the STARTBR command, to identify the start of the browse, and terminated with the ENDBR command.

You must provide, in the RIDFLD option, a data area that is sufficiently large to contain a complete identifier (full key, RBA, or RRN) of records in the file. This data area is used both as an output and as an input parameter.

It is used as an output parameter when CICS, on completion of each READPREV command, places the complete identifier of the record just retrieved into the RIDFLD data area. CICS then holds this identifier to mark the point from which the subsequent READPREV is to continue.

It may also be used as an input parameter. Modifying the RIDFLD before issuing the next READPREV command causes that command to reposition the browse to the new identifier, from which it continues in the usual way. The modified record identifier must always be a full key, RBA, or RRN. A generic key may not be specified, nor may a browse that was started with the GENERIC option include a READPREV command.

If you include a READPREV command immediately following a STARTBR command, your STARTBR command RIDFLD must specify the key of a record that exists on the data set; otherwise the NOTFND condition will occur.

A READPREV command following a READNEXT, or a STARTBR or RESETBR that did not specify a 'last' key value, is treated as though the RIDFLD value had been modified and results in a reposition (as above).

Reading files accessed in RLS mode

For files accessed in RLS mode, you can include the UPDATE keyword on the READPREV request to update some records during the browse. If you specify UPDATE you must also specify TOKEN. You can then update the record by issuing a DELETE or REWRITE command that specifies the TOKEN returned on the browse function.

Note: TOKEN without the UPDATE keyword implies UPDATE.

Use of the UPDATE option is subject to the following rules:

- You can specify UPDATE on a READPREV command only if the file is accessed in RLS mode. If you specify UPDATE for a file accessed in non-RLS mode, CICS returns INVREQ.
- You can specify UPDATE on READPREV, but not on the STARTBR or RESETBR commands.
- You can intermix UPDATE and non-update requests within the same browse.
- CICS does not preserve the UPDATE option for you from one READPREV command to the next.

  CICS supports only one TOKEN in a browse sequence, and the TOKEN value on each READPREV invalidates the previous value.

Locks for UPDATE

Specifying UPDATE on a READPREV command acquires an exclusive lock. The duration of these exclusive locks within a browse depends on the action your application program takes:

- If you decide to DELETE or REWRITE the last record acquired by a READPREV UPDATE in a browse, using the associated token, the lock remains active as follows:
  - If the file is recoverable, the lock is released at completion of the next sync point or rollback.
  - If the file is non-recoverable, the lock will be released by the time ENDBR has completed, but might be released earlier.
- If you decide *not* to update the last record read, CICS frees the exclusive lock when your program either issues another READNEXT or READPREV command, or ends the browse.

Important: The UNLOCK command does *not* free an exclusive lock held by VSAM against a record acquired by READPREV UPDATE. An UNLOCK in a browse invalidates the TOKEN returned by the last request.

Locks for read integrity

Specifying one of the read integrity options acquires a shared lock on each READPREV. The duration of these shared locks with a browse depends on the type of read integrity you specify:

- If you specify CONSISTENT read integrity, the shared lock is held only for the duration of each read request—that is, until the record is returned to your program.
- If you specify REPEATABLE read integrity, the shared lock is held for the duration of the unit of work in which the browse is performed. In this case, your program could acquire a large number of shared locks, which will prevent the granting of exclusive locks for update functions. You should use REPEATABLE read integrity in a browse with caution.

Function shipping READPREV with UPDATE or read integrity

If a READPREV command that specifies UPDATE or one of the read integrity options is function-shipped to a member of the CICS family of products that does not support UPDATE or the read integrity options, the request fails:

- If an ISC link is used, the request receives an ATNI abend.
- If an MRO link is used, the request receives an AXF8 abend.

  AXF8 is an abend code, received by the sending side of a function-shipped request. It indicates that an attempt has been made to send a request that specifies UPDATE on an MRO link to a CICS region that does not support update or read integrity options.

Read integrity

When a file is accessed in RLS mode, non-update read requests can specify read integrity options: UNCOMMITTED, CONSISTENT, or REPEATABLE.

If you do not specify any of these keywords, CICS uses the value specified on the READINTEG parameter of the FILE resource definition, for which the default is UNCOMMITTED.

If you want to use the level of read integrity specified in the READINTEG keyword of the FILE definition, and then you need to change from using a local file to a remote file, or if you change the location of a remote file, ensure that:

- The remote file-owning region is at the CICS Transaction Server for OS/390, Version 1 Release 1 (or later) level.
- The FILE definition in the remote system specifies:
  - RLS mode
  - The correct read integrity values for your application.

Retained and active locks

RECORDBUSY refers to active locks and LOCKED refers to retained locks.

These locks affect READPREV requests which acquire locks; that is, update requests and requests with read integrity. These are the kinds of READPREV requests which are referred to in the following bullets. Other READPREV requests are unaffected by the presence of retained or active locks.

- READPREV requests for records that have *retained* locks are always rejected with a LOCKED response.
- READPREV requests for records that have *active* locks wait for the lock to be released, except when the NOSUSPEND keyword is specified, in which case CICS returns the RECORDBUSY response.

Options

CONSISTENT (RLS only)
The record is read with a level of read integrity provided by a VSAM shared lock that lasts for the duration of the request.

If the record is being modified by another task, which therefore holds an exclusive lock, the READPREV request waits until the update is complete (unless NOSUSPEND is also specified) as follows:

- For a READPREV request against a non-recoverable file, the READPREV completes as soon as any VSAM request performing the update completes.
- For a READPREV request against a recoverable file, the READPREV request completes when the updating task completes its next syncpoint or rollback.

FILE(*filename*)
specifies the name of the file being browsed.

If SYSID is specified, the data set to which this file refers is assumed to be on a remote system irrespective of whether the name is defined in the file resource definition. Otherwise, the file definition is used to find out whether the data set is on a local or a remote system.

INTO(*data-area*)
specifies the data area into which the record retrieved from the data set is to be written.

KEYLENGTH(*data-value*)
specifies the length (halfword binary) of the key that has been specified in the RIDFLD option, except when RBA or RRN is specified, in which case KEYLENGTH is not valid. If the length specified is different from the length defined for the data set, the INVREQ condition occurs.

For remote files, the KEYLENGTH can be specified in the FILE definition. If KEYLENGTH is not defined there, and is not specified in the application program, and the key is longer than 4 characters, the default value is 4.

LENGTH(*data-area*)
specifies the length, as a halfword binary value, of the data area where the record is to be put. On completion of the READNEXT command, the LENGTH parameter contains the actual length of the record.

This option must be specified if SYSID is specified.

If the file is on a remote system, the LENGTH parameter need not be set here but must be set in the file resource definition

If the file is on a local system, the LENGTH parameter must be set for variable-length records, using the INTO option, but not for fixed-length records. It is, however, advisable to specify the length for fixed-length records because:

- It causes a check to be made that the record being read is not too long for the available data area.
- When browsing fixed-length records into an area longer than the record being accessed, the LENGERR condition is raised for COBOL, C, PL/I, and

assembler-language applications if the LENGTH option is specified. If the LENGTH option is not explicitly specified, then the LENGERR condition is returned if the length can be inferred from the data area. If the length specified exceeds the file record length, CICS uses the longer length for the move. If the target area in the application program is not large enough, storage is overlaid beyond the target area.

If you specify the SET option, you need not also specify the LENGTH option.

When browsing into a target data area longer than the record being read, the contents of the target data area, from the end of the retrieved record to the end of the target data area, are unpredictable.

If you specify the INTO option, the LENGTH argument must be a data area that specifies the largest record the program accepts. If the retrieved record is longer than the value specified in the LENGTH option, the record is truncated to the specified value and the LENGERR condition occurs. In this case, the LENGTH data area is set to the length of the record before truncation.

Note that a file control command issued against a variable-length record in a file defined on the local CICS system fails with a LENGERR condition if a length is not specified. However, if the same command is issued against a file defined on a remote system, the command does not fail.

NOSUSPEND (RLS only)
The request does not wait if the record is locked by VSAM with an active lock, including records locked as the result of a DEADLOCK.

Note: Requests that specify NOSUSPEND wait for at least 1 second before CICS returns the RECORDBUSY response.

RBA
(VSAM KSDS or ESDS base data sets, or CICS-maintained data tables only, but not paths) specifies that the record identification field specified in the RIDFLD option contains a relative byte address.

This option must be specified when the STARTBR or RESETBR command specified the RBA option. It must not be specified when the STARTBR or RESETBR command did not specify RBA.

You cannot use RBA for:

- User-maintained data tables
- Coupling facility data tables
- Any KSDS file opened in RLS access mode
- KSDS files that use extended addressing.

Also, you are recommended not to use RBA for ESDS files that hold more than 4GB. (Use XRBA instead.)

REPEATABLE (RLS only)
The record is read with a level of read integrity provided by a VSAM shared lock that lasts for the duration of the unit of work in which the read request is issued.

If the record is being modified by another task, which therefore holds an exclusive lock, the READPREV request waits until the update is complete (unless NOSUSPEND is also specified) as follows:

- For a recoverable file, the READPREV request completes when the updating transaction completes its next sync point or rollback.

- For a non-recoverable file, the READPREV completes as soon as the VSAM request performing the update completes.

After the READPREV request has completed, the record remains locked to the task that issued the READPREV. Other tasks can continue to read the record, but no other task is allowed to update the record until the task that issued the READPREV performs its next sync point or rollback.

REQID(*data-value***)**
specifies as a halfword binary value a unique request identifier for a browse, used to control multiple browse operations on a file. If this option is not specified, a default value of zero is assumed.

RIDFLD(*data-area***)**
specifies the record identification field. The contents can be a key, a relative byte address, or relative record number. For a relative byte address or a relative record number, the format of this field must be fullword binary. For a relative byte address, the RIDFLD value can be greater than or equal to zero. For a relative record number, the RIDFLD value can be greater than or equal to 1.

On completion of the READPREV command, this field is updated by CICS with the complete identification of the record retrieved.

RRN
(VSAM RRDS) specifies that the record identification field specified in the RIDFLD option contains a relative record number.

SET(*ptr-ref***)**
specifies the pointer reference that is to be set to the address of the retrieved record.

In assembler language, if the DUPKEY condition occurs, the register specified will not have been set, but can be loaded from DFHEITP1.

The pointer reference is valid until the next READPREV or READNEXT command that specifies SET for the same browse (REQID) for the same file. The pointer is no longer valid after an ENDBR or SYNCPOINT command. If you want to retain the data within the field addressed by the pointer, you must move it to your own area.

If DATALOCATION(ANY) is associated with the application program, the address returned in the SET pointer can be above or below the 16 MB line.

If DATALOCATION(BELOW) is associated with the application program, the address returned in the SET pointer is below the 16 MB line.

If TASKDATAKEY(USER) is specified for the running task, the data returned is in user-key storage; otherwise it is in CICS-key storage.

SYSID(*systemname***)**
specifies the name of the system to which the request is directed.

If you specify SYSID, and omit both RBA and RRN, you must also specify LENGTH and KEYLENGTH.

TOKEN(data-area) (RLS only)
Returns, as a fullword binary value, a unique identifier for this READPREV UPDATE request. This is an output value returned by file control to the requesting task, for use in associating a subsequent REWRITE or DELETE (or UNLOCK) request with the record returned on this READPREV command.

Your application program must specify the returned TOKEN on a subsequent REWRITE or DELETE command to identify the record being rewritten or deleted. Your application program can also specify the returned TOKEN on a subsequent UNLOCK command to identify the token that is being invalidated.

You must specify TOKEN if you specify UPDATE (but UPDATE is assumed if you specify TOKEN without UPDATE).

CICS supports only one active TOKEN at a time for a given REQID. Thus, a TOKEN value remains valid only until the following READNEXT, READPREV, or ENDBR command for this browse, or until a REWRITE, DELETE, or UNLOCK command (whichever is first).

TOKEN can be function shipped. However, if a request specifying TOKEN is function shipped to a member of the CICS family of products that does not recognize this keyword, the request fails.

UNCOMMITTED
The record is read without read integrity. The current data, as known to VSAM, is returned. No attempt is made to serialize this read request with any concurrent update activity for the same record. The record may be being updated by another transaction, therefore the value of the record may change later if that update is subsequently backed out.

UPDATE (RLS only)
Specifies that the record is to be obtained for updating or deletion. If this option and TOKEN are both omitted, read only is assumed.

If you specify UPDATE, you must also specify TOKEN.

UPDATE is only valid for files defined to the local region

XRBA
specifies that the record identification field specified in the RIDFLD option contains an extended relative byte address. This option should be used when browsing records in an ESDS extended addressing data set.

You cannot specify XRBA on a READPREV command unless the associated STARTBR or RESETBR command also specified XRBA.

KSDS data sets cannot be accessed by XRBA.

Conditions

15 DUPKEY
RESP2 values:

140 A record is retrieved by way of an alternate index with the NONUNIQUEKEY attribute, and another alternate index record with the same key exists.

In assembler language, if the SET option is being used, the specified register has not been set, but can be loaded from DFHEITP1.

Default action: terminate the task abnormally.

20 ENDFILE
RESP2 values:

90 An end-of-file condition is detected during a browse.

Default action: terminate the task abnormally.

12 FILENOTFOUND
RESP2 values:

1 A file name referred to in the FILE option is not defined to CICS.

Default action: terminate the task abnormally.

21 ILLOGIC
The browse that is currently in progress is terminated when this condition is raised.

RESP2 values (VSAM):

110 A VSAM error occurs that is not in one of the other CICS response categories.

See EIBRCODE in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

20 The FILE definition does not allow updates.

24 A READPREV command is issued for a file for which the previous STARTBR or RESETBR command has the GENERIC option.

26 The KEYLENGTH option is specified and the specified length does not equal the length defined for the data set for this file refers to.

37 The type of record identification (for example, key or relative byte address) used to access a data set during the browse has been changed. You cannot specify one type of addressing on STARTBR and another on READPREV.

39 A READPREV is issued for a BDAM file.

41 The REQID, SYSID, or file name does not match that of any successful STARTBR command.

52 CONSISTENT is not allowed because the file is not a VSAM file that is accessed in RLS mode.

53 REPEATABLE is not allowed because the file is not a VSAM file that is accessed in RLS mode.

54 UPDATE is not allowed because the file is not a VSAM file that is accessed in RLS mode.

55 NOSUSPEND is not allowed because the file is not a VSAM file that is accessed in RLS mode.

57 An attempt to update the file has failed because the associated VSAM data set has an AVAILABILITY state of RREPL, which disallows update operations except those from replication programs.

Default action: terminate the task abnormally.

17 IOERR
RESP2 values:

120 An I/O error occurred during the browse. An I/O error is any unusual event that is not covered by a CICS condition.

For VSAM files, IOERR usually indicates a hardware error. Further information is available in the EXEC interface block; for details, see EIB fields.

For a coupling facility data table, an IOERR indicates a bad response returned from a coupling facility access.

Default action: terminate the task abnormally.

54 ISCINVREQ

RESP2 values:

70 The remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

22 LENGERR

RESP2 values:

10 Neither the LENGTH nor the SET option is specified for a file with variable-length records.

11 The length of the record read with the INTO option specified exceeds the value specified in the LENGTH option; the record is truncated, and the data area supplied in the LENGTH option is set to the actual length of the record.

13 An incorrect length is specified for a file with fixed-length records.

Default action: terminate the task abnormally.

100 LOCKED

RESP2 values:

106 The read request specifies the UPDATE keyword, or one of the read integrity keywords CONSISTENT or REPEATABLE, or the file resource definition specifies read integrity, but VSAM holds a retained lock against the record (see "Retained and active locks" on page 398).

The key of the locked record is not returned to your application program. If you handle this condition and control is returned to your program, the browse can continue and retrieve the record following the locked record by issuing another READPREV request.

The LOCKED condition also can occur for a request to a recoverable CFDT that uses the locking model, if the record that is being read is locked by a retained lock. See Coupling facility data table retained locks for information about investigating retained locks on records in a coupling facility data table.

Default action: abend the task with code AEX8.

70 NOTAUTH

RESP2 values:

101 A resource security check has failed on FILE(filename).

Default action: terminate the task abnormally.

13 NOTFND

RESP2 values:

80 An attempt to retrieve a record based on the search argument provided is unsuccessful. One place where this may occur is where the READPREV command immediately follows a STARTBR or RESETBR command that specified GTEQ and the key of a record that does not exist on the data set.

81 XRBA was specified, and the value of RIDFLD was greater than 4 GB, but the data set is not an extended addressing ESDS.

Default action: terminate the task abnormally.

101 RECORDBUSY
RESP2 values:

107 NOSUSPEND is specified on the request but VSAM holds an active lock against the record, which would cause the request to wait (see "Retained and active locks" on page 398).

The key of the locked record is not returned to your application program. If you handle this condition and control is returned to your program, the browse can continue and retrieve the record following the locked record by issuing another READPREV request.

Default action: abend the task with code AEX9.

53 SYSIDERR
RESP2 values:

130 The SYSID option specifies a name that is neither the local system nor a remote system that is defined by a CONNECTION or IPCONN definition. SYSIDERR also occurs when the link to the remote system is known but unavailable. In the case of an IPCONN, SYSIDERR occurs if the link is known but either the local or remote systems do not support file control commands that are function shipped using IP interconnectivity.

131 For a coupling facility data table, the connection to the coupling facility data table server has failed. This could be because the server itself has failed, or the server is available, but CICS has failed to connect to it.

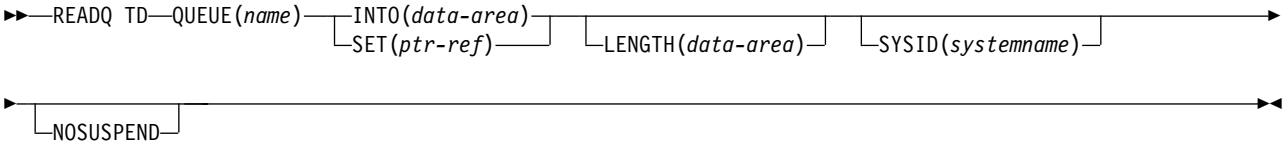
132 The READPREV is issued against a coupling facility data table that no longer exists, probably because of a coupling facility failure, in which case the coupling facility data table server also fails. See Setting up and running a coupling facility data table server for information about restarting a coupling facility data table server and reloading a table.

Default action: terminate the task abnormally.

READQ TD

Read data from the transient data queue.

READQ TD



Conditions: DISABLED, INVREQ, IOERR, ISCINVREQ, LENGERR, LOCKED, NOTAUTH, NOTOPEN, QBUSY, QIDERR, QZERO, SYSIDERR

This command is threadsafe when it is used with a queue in a local CICS region, or function shipped to a remote CICS region over an IPIC connection. It is non-threadsafe when it is function shipped to a remote CICS region over another type of connection.

Description

READQ TD reads transient data from a queue (after which the record is no longer available).

If you are using automatic transaction initiation (ATI), your application should test for the QZERO condition to ensure that termination of an automatically initiated task occurs only when the queue is empty. For introductory information about ATI, see Automatic transaction initiation (ATI).

If the **READQ TD** command attempts to access a record in a logically recoverable intrapartition queue that is being written to, or deleted by, another task, and there are no more committed records, the command waits until the queue is no longer being used for output. If, however, the NOSUSPEND option has been specified, or there is an active HANDLE CONDITION for QBUSY, the QBUSY condition is raised.

Options

INTO(*data-area***)**
Specifies the user data area into which the data read from the transient data queue is to be placed.

LENGTH(*data-area***)**
Specifies the length, as a halfword binary value, of the record to be read.

If you specify the INTO option, LENGTH specifies the maximum length of data that the program accepts. If the value specified is zero or less than zero, a LENGERR condition will be raised. If the length of the data exceeds the value specified, the data is truncated to that value and the LENGERR condition occurs. On completion of the retrieval operation, the data area is set to the original length of the data record read from the queue.

If you specify the INTO option, LENGTH need not be specified if the length can be generated by the compiler from the INTO variable. See “LENGTH options in CICS commands” on page 10 for more information about when LENGTH must be specified.

NOSUSPEND
Specifies that if the application program attempts to read from a queue that is already being used for output, the task is not suspended until the queue becomes available. Instead, the QBUSY condition is raised.

Note, however, that if a HANDLE CONDITION for QBUSY is active when the command is executed, this also overrides the default action, and control is passed to the user label supplied in the HANDLE CONDITION. This takes precedence over the NOSUSPEND option but is, of course, negated by either NOHANDLE or RESP.

This option only applies to intrapartition queues.

QUEUE(*name***)**
Specifies the symbolic name (1 - 4 alphanumeric characters) of the queue to be read from. The named queue must have been defined to CICS.

If SYSID is specified, the queue is assumed to be on a remote system whether or not it is defined as remote. Otherwise the transient data queue definition is used to find out whether the data set is on a local or a remote system.

SET(*ptr-ref***)**
Specifies a pointer reference that is to be set to the address of the data read from the queue. CICS acquires an area large enough to hold the record and sets the pointer reference to the address of that area. The area is retained until another transient data command is executed. The pointer reference, unless changed by other commands or statements, is valid until the next **READQ TD** command or the end of task.

If DATALOCATION(ANY) is associated with the application program, the address of the data can be above or below the 16 MB line.

If DATALOCATION(BELOW) is associated with the application program, and the data resides above the 16 MB line, the data is copied below the 16 MB line, and the address of this copy is returned.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

SYSID(*systemname***)**
(remote systems only) Specifies the name (1 - 4 characters) of the system to which the request is directed.

Conditions

84 DISABLED
occurs when the queue has been disabled.

Default action: terminate the task abnormally.

16 INVREQ
Occurs if **READQ** names an extrapartition queue that has been opened for output. This condition cannot occur for intrapartition queues.

Default action: terminate the task abnormally.

17 IOERR
Occurs when an input/output error occurs and the data record in error is skipped.

This condition occurs if the queue can be read; when the queue cannot be read, a QZERO condition occurs in this situation.

This condition can also occur if the FREE=CLOSE operand has been used in the data set definition for an extrapartition queue, and the queue has been closed and reopened.

Default action: terminate the task abnormally.

54 ISCINVREQ
Occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

22 LENGERR
Occurs if **READQ** names an INTO area that cannot hold all the data that is to be returned to the application or an invalid length has been supplied. The check is made after the XTDIN exit has been invoked.

Default action: terminate the task abnormally.

100 LOCKED
Occurs when the request cannot be performed because use of the queue has been restricted owing to a unit of work failing indoubt. This can happen on any request for a logically-recoverable queue defined with WAIT(YES) and WAITACTION(REJECT) in the TDQUEUE resource definition.

Specify WAIT(YES) and WAITACTION(QUEUE) in the TDQUEUE resource definition if you want the transaction to wait.

Default action: terminate the task abnormally.

70 NOTAUTH
Occurs when a resource security check has failed on QUEUE(name).

Default action: terminate the task abnormally.

19 NOTOPEN
Occurs if the destination is closed. This condition applies to extrapartition queues only.

Default action: terminate the task abnormally.

25 QBUSY
Occurs if a **READQ TD** command attempts to access a record in a logically recoverable intrapartition queue that is being written to or is being deleted by another task, and there are no more committed records.

The NOSUSPEND option must be specified, or a HANDLE for the condition must be active, for this condition to be raised.

This condition applies only to intrapartition queues.

Default action: ignore the condition.

44 QIDERR
Occurs if the symbolic destination to be used with **READQ TD** cannot be found.

Default action: terminate the task abnormally.

23 QZERO
Occurs when the destination (queue) is empty or the end of the queue has been reached.

Default action: terminate the task abnormally.

53 SYSIDERR
Occurs when the SYSID option specifies a name that is neither the local system nor a remote system (made known to CICS by defining a CONNECTION or an IPCONN). SYSIDERR also occurs when the link to the remote system is closed.

Default action: terminate the task abnormally.

Examples

The following example shows how to read a record from an intrapartition data set (queue), which in this case is the control system message log (CSML), into a data area specified in the request:

```
EXEC CICS READQ TD
     QUEUE('CSML')
     INTO(DATA)
     LENGTH(LDATA)
```

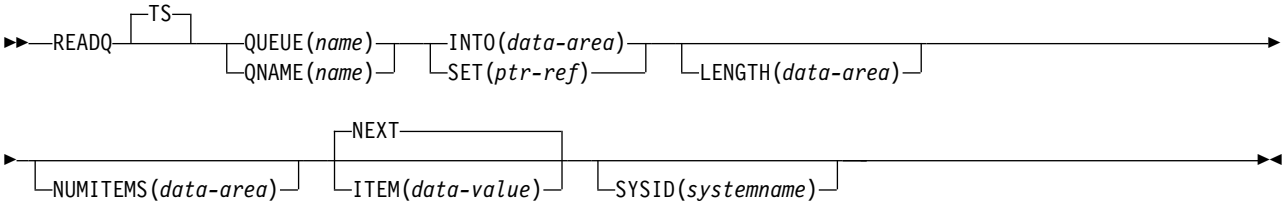
The following example shows how to read a record from an extrapartition data set (queue) that has fixed-length records into a data area provided by CICS; the pointer reference specified by the SET option is set to the address of the storage area reserved for the data record. It is assumed that the record length is known.

```
EXEC CICS READQ TD
     QUEUE(EX1)
     SET(PREF)
```

READQ TS

Read data from a temporary storage queue.

READQ TS



Conditions: INVREQ, IOERR, ISCINVREQ, ITEMERR, LENGERR, NOTAUTH, QIDERR, SYSIDERR

This command is threadsafe when it is used with a queue in main storage or auxiliary storage, either in a local CICS region, or function shipped to a remote CICS region over an IPIC connection. It is also threadsafe when it is used with a queue in a shared temporary storage pool in a z/OS coupling facility that is managed by a temporary storage data sharing server (TS server). The command is non-threadsafe when it is function shipped to a remote CICS region over another type of connection other than IPIC.

Note for dynamic transaction routing: Using this command might create inter-transaction affinities that adversely affect the use of dynamic transaction routing. For more information about transaction affinities, see Affinity .

Description

READQ TS retrieves data from a temporary storage queue in main or auxiliary storage.

Options

INTO(*data-area***)**
Specifies the data area into which the data is to be written. The data area can be any variable, array, or structure.

ITEM(*data-value***)**
Provides a halfword binary value that specifies the item number of the logical record to be retrieved from the queue.

LENGTH(*data-area***)**
Specifies the length, as a halfword binary value, of the record to be read.

If you specify the INTO option, LENGTH need not be specified if the length can be generated by the compiler from the INTO variable.

If you specify INTO, LENGTH defines the maximum length of data that the program accepts. If the value specified is less than zero, zero is assumed. If the length of the data exceeds the value that is specified, the data is truncated to that value and the LENGERR condition occurs.

On completion of the retrieval operation, the data area is set to the original length of the data record that is read from the queue.

If you specify the SET option, the LENGTH must be specified.

For more information about when LENGTH must be specified, see "LENGTH options in CICS commands" on page 10.

NEXT
Specifies retrieval for the next sequential logical record following the last record that was retrieved by any task, or the first record if no previous record has been retrieved.

Attention: It is possible for two tasks to interleave if they lose control during the browse operation. For example, task 1 might retrieve items 1, 3, and 6 while task 2 retrieves items 2, 4, and 5. Using the **READQ TS** command with NEXT from a threadsafe program increases the likelihood of tasks interleaving, because they are running in parallel on their own TCBs. If the order of retrieval of items is important, add serialization logic to the application in order to single thread the browse of the queue, especially if the application is to be defined with CONCURRENCY(THREADSAFE) or CONCURRENCY(REQUIRED).

NUMITEMS(*data-area***)**
Specifies a halfword binary field into which CICS stores a number that indicates how many items there are in the queue. This occurs only if the command completes normally.

QNAME(*name***)**
An alternative to QUEUE, QNAME specifies the symbolic name (1 - 16 characters) of the queue to be read from. If the name has fewer than 16 characters, you must still use a 16-character field, padded with blanks if necessary.

QUEUE(*name***)**
Specifies the symbolic name (1 - 8 characters) of the queue to be read from. If the name has fewer than 8 characters, you must still use an 8-character field, padded with blanks if necessary.

SET(*ptr-ref***)**
Specifies the pointer reference that is set to the address of the retrieved data. The pointer reference, unless changed by other commands or statements, is valid until the next READQ TS command or the end of task.

If the application program is defined with DATALOCATION(ANY), the address of the data can be above or below the 16 MB line. If the application program is defined with DATALOCATION(BELOW), the address of the data is below the 16 MB line.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data that is returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data that is returned is in a CICS-key.

SYSID(*systemname***)**
(Remote and shared queues only) Specifies the system name (1 - 4 characters) identifying the remote system or shared queue pool to which the request is directed. TSMODEL resource definitions do not support specifying a SYSID for

a queue that resides in a temporary storage data sharing pool. Use the QUEUE or QNAME option instead. Using an explicit SYSID for a shared queue pool requires the support of a temporary storage table (TST).

Conditions

16 INVREQ
Occurs in either of the following situations:

- The queue was created by CICS internal code.
- The queue name that is specified consists solely of binary zeros.

Default action: terminate the task abnormally.

17 IOERR
RESP2 values:

5 There is an unrecoverable input/output error for a shared queue.

Default action: terminate the task abnormally.

54 ISCINVREQ
Occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

26 ITEMERR
Occurs in either of the following situations:

- The item number that is specified is invalid (that is, outside the range of item numbers that are written to the queue).
- An attempt is made to read beyond the end of the queue by using the NEXT (default) option.

Default action: terminate the task abnormally.

22 LENGERR
Occurs when the length of the stored data is greater than the value specified by the LENGTH option.

This condition applies only to the INTO option and cannot occur with SET.

Default action: terminate the task abnormally.

70 NOTAUTH
RESP2 values:

101 A resource security check has failed on QUEUE(name).

Default action: terminate the task abnormally.

44 QIDERR
Occurs when the queue specified cannot be found, either in main or in auxiliary storage.

Default action: terminate the task abnormally.

53 SYSIDERR
RESP2 values:

4 Occurs in any of the following situations:

- The SYSID option specifies a name that is not the local system or a remote system (made known to CICS by defining a CONNECTION or an IPCONN).

- When IPIC connectivity is used, the local system, the remote system, or both, are not CICS TS 4.2 or later regions.
- The link to the remote system is closed.
- The CICS region in which the temporary storage command is executed fails to connect to the TS server that manages the TS pool that supports the referenced temporary storage queue. For example, this situation can occur if the CICS region is not authorized to access the temporary storage server.

  This condition can also occur if the temporary storage server is not started, or because the server has failed (or been stopped) while CICS continues to run.

Default action: terminate the task abnormally.

Examples

The following example shows how to read the first (or only) record from a temporary storage queue into a data area that is specified in the request. The LENGTH data area is given the value of the length of the record.

```
EXEC CICS READQ TS
     ITEM(1)
     QUEUE(UNIQNAME)
     INTO(DATA)
     LENGTH(LDATA)
```

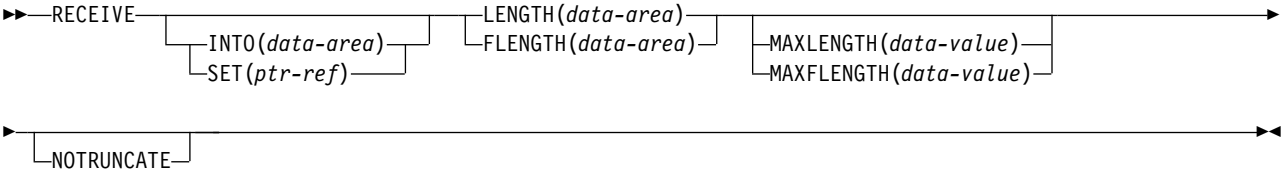
The following example shows how to read the next record from a temporary storage queue into a data area that is provided by CICS. The pointer reference that is specified by the SET option is set to the address of the storage area that is reserved for the data record, and the LENGTH data area is given the value of the length of the record.

```
EXEC CICS READQ TS
     QUEUE(DESCRQ  )
     SET(PREF)
     LENGTH(LENG)
     NEXT
```

RECEIVE (z/OS Communications Server default)

Receive data from standard CICS terminal support or from a task that is not attached to a terminal.

RECEIVE (default)



Conditions: INVREQ, LENGERR, NOTALLOC

Description

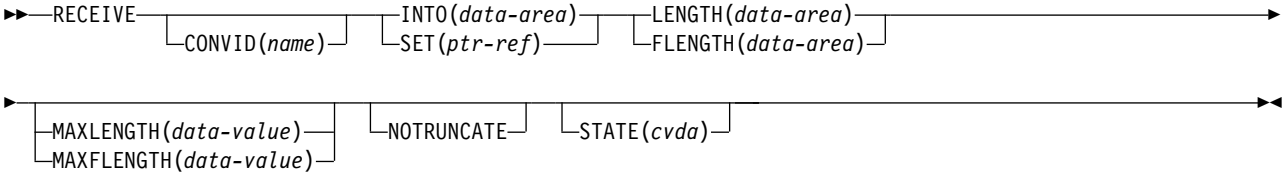
This form of the RECEIVE command is used by all CICS-supported terminals for which the other RECEIVE descriptions are not appropriate.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

RECEIVE (APPC)

Receive data on an APPC mapped conversation.

RECEIVE (APPC)



Conditions: EOC, INVREQ, LENGERR, NOTALLOC, SIGNAL, TERMERR

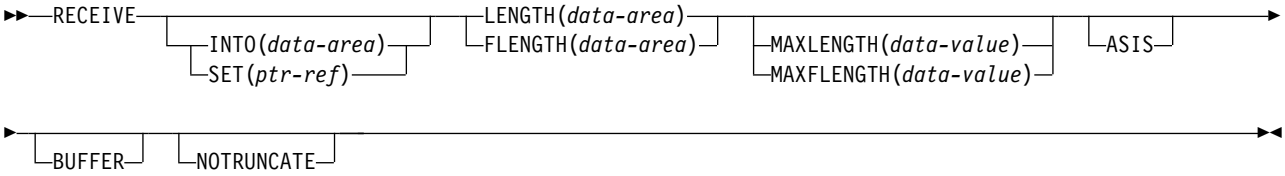
Description

RECEIVE receives data from the conversation partner in an APPC mapped conversation.

RECEIVE (LUTYPE2/LUTYPE3)

Receive data from a 3270-display logical unit (LUTYPE2) or a 3270-printer logical unit (LUTYPE3).

RECEIVE (LUTYPE2/LUTYPE3)



Conditions: EOC, INVREQ, LENGERR, TERMERR

Description

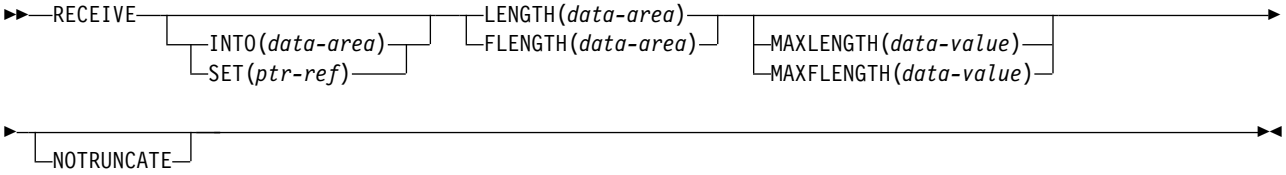
RECEIVE receives data from the terminal.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) (and BUFFER has not been specified), you can omit both the INTO and SET options.

RECEIVE (LUTYPE4)

Receive data from an LUTYPE4 logical unit.

RECEIVE (LUTYPE4)



Conditions: EOC, EODS, INBFMH, INVREQ, LENGERR, SIGNAL, TERMERR

Description

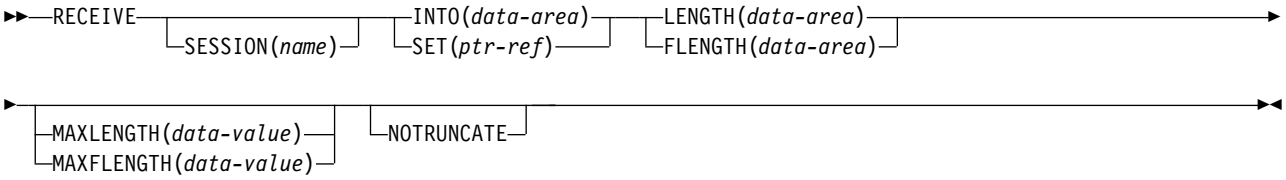
RECEIVE receives data from the terminal.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

RECEIVE (LUTYPE6.1)

Receive data on an LUTYPE6.1 session.

RECEIVE (LUTYPE6.1)



Conditions: EOC, INBFMH, INVREQ, LENGERR, NOTALLOC, SIGNAL, TERMERR

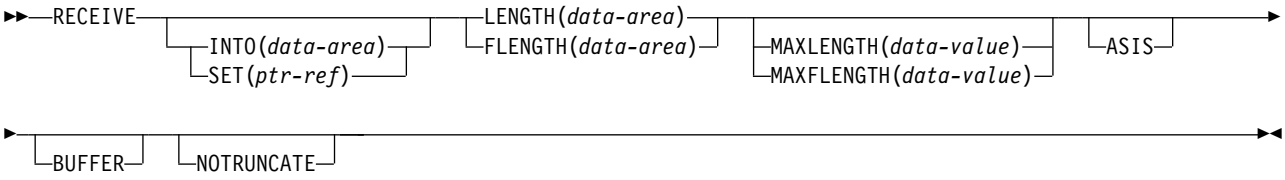
Description

RECEIVE receives data from the conversation partner in an LUTYPE6.1 conversation.

RECEIVE (3270 logical)

Receive data from a 3270 logical unit.

RECEIVE (3270 logical)



Conditions: INVREQ, LENGERR, TERMERR

Description

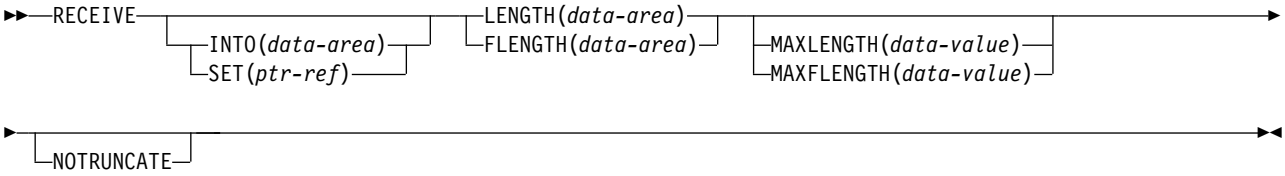
RECEIVE receives data from a terminal.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) (and BUFFER has not been specified), you can omit both the INTO and SET options.

RECEIVE (3600 pipeline)

Receive initial input data from a 3600 pipeline logical unit. Subsequent RECEIVEs for further input data are not allowed.

RECEIVE (3600 pipeline)



Conditions: INVREQ, LENGERR, NOTALLOC

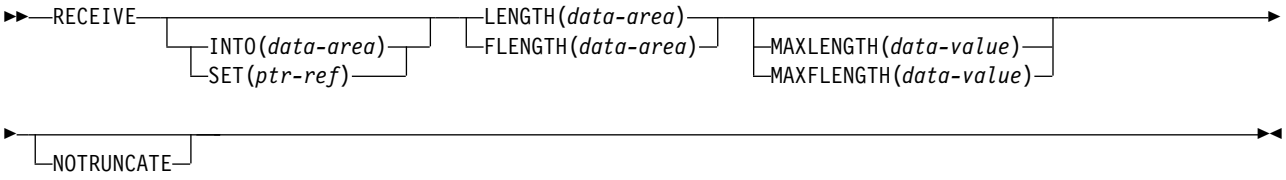
Description

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

RECEIVE (3600-3601)

Receive data from a 3600 (3601) logical unit.

RECEIVE (3600-3601)



Conditions: EOC, EODS, INBFMH, INVREQ, LENGERR, SIGNAL, TERMERR

Description

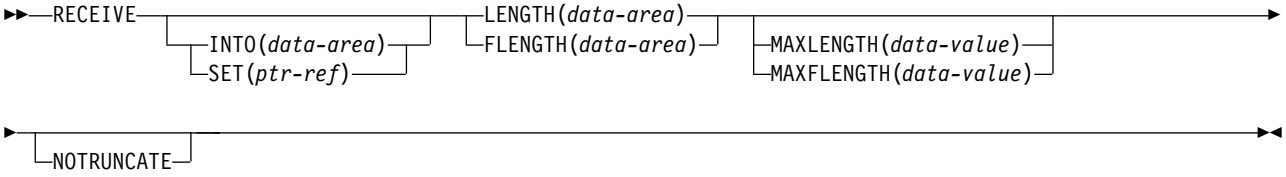
RECEIVE receives data from the terminal. This form of RECEIVE also applies to the 3630 plant communication system.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

RECEIVE (3600-3614)

Receive data from a 3600 (3614) logical unit.

RECEIVE (3600-3614)



Conditions: INVREQ, LENGERR, TERMERR

Description

RECEIVE receives data from the terminal.

The data-stream and communication format used between a CICS application program and a 3614 is determined by the 3614. The application program is therefore device dependent when handling 3614 communication.

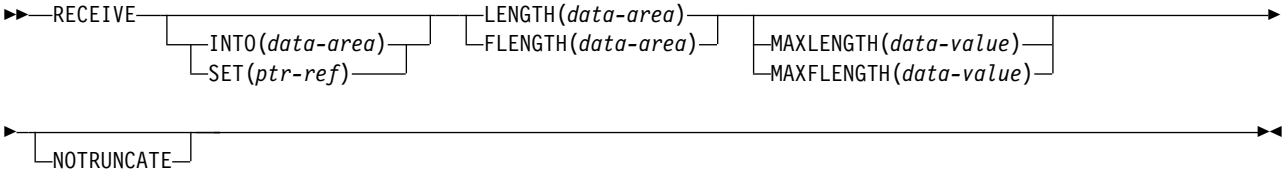
For further information about designing 3614 application programs for CICS, refer to the *IBM 4700/3600/3630 Guide*.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

RECEIVE (3650)

Receive data from 3650 logical units.

RECEIVE (3650)



Conditions: EOC, EODS, INBFMH, INVREQ, LENGERR, TERMERR

Description

RECEIVE receives data from the terminal. This form of RECEIVE also applies to the following 3650 devices:

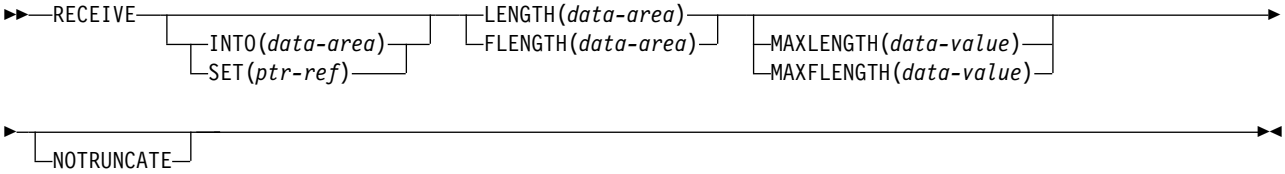
- Interpreter logical unit
- Host conversational (3270) logical unit
- Host conversational (3653) logical unit
- 3650/3680 command processor logical unit

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

RECEIVE (3767)

Receive data from a 3767 interactive logical unit.

RECEIVE (3767)



Conditions: EOC, INVREQ, LENGERR, SIGNAL, TERMERR

Description

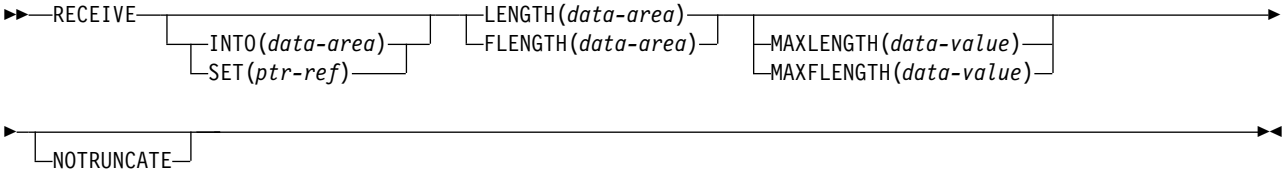
RECEIVE receives data from the terminal. This form of RECEIVE also applies to the 3770 interactive logical unit.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

RECEIVE (3770)

Receive data from a 3770 batch logical unit.

RECEIVE (3770)



Conditions: EOC, EODS, INBFMH, INVREQ, LENGERR, SIGNAL, TERMERR

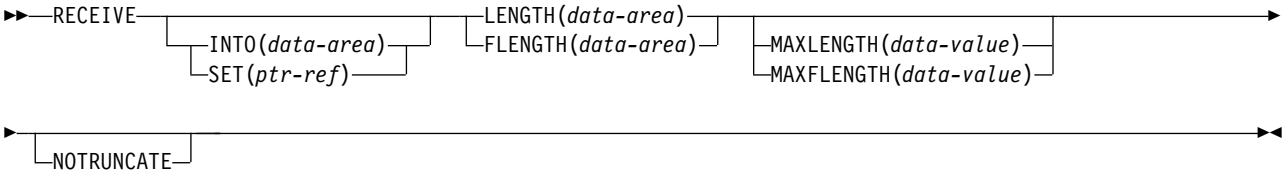
Description

RECEIVE receives data from the terminal.If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

RECEIVE (3790 full-function or inquiry)

Receive data from a 3790 full-function or inquiry logical unit.

RECEIVE (3790 full-function or inquiry)



Conditions: EOC, EODS, INBFMH, INVREQ, LENGERR, SIGNAL, TERMERR

Description

RECEIVE receives data from the terminal. This form of RECEIVE also applies to the following devices:

- 3650/3680 full-function logical unit
- 3770 full-function logical unit

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

RECEIVE: z/OS Communications Server options

Common options used on the RECEIVE command (z/OS Communications Server).

Options

ASIS
specifies that lowercase characters in the 3270 input data stream are not translated to uppercase; this allows the current task to receive a message containing both uppercase and lowercase data.

This option has no effect on the first RECEIVE command of a transaction, because terminal control performs a READ INITIAL and uses the terminal defaults to translate the operation data.

This option has no effect if the screen contains data before a transaction being initiated. This data is read and translated in preparation for the next task and the first RECEIVE command in that task retrieves the translated data.

Note: If you are using a katakana terminal, you might see some messages containing mixed English and katakana characters. That is because katakana terminals cannot display mixed-case output. Uppercase characters in the data stream appear as uppercase English characters, but lowercase characters appear as katakana characters. If this happens, ask your system programmer to specify MSGCASE=UPPER in the system initialization parameters so that messages contain uppercase characters only.

BUFFER
specifies that the contents of the 3270 buffer are to be read, beginning at buffer location 1 and continuing until all contents of the buffer have been read. All character and attribute sequences (including nulls) appear in the input data stream in the same order as they appear in the 3270 buffer.

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

For compatibility with earlier releases, SESSION is accepted as a synonym for CONVID. New programs should use CONVID.

If this option is omitted, the principal facility is assumed.

FLENGTH(*data-area***)**
An alternative to LENGTH. For architectural reasons, this option is limited to a maximum of 32K for all terminal-related RECEIVE commands.

INTO(*data-area***)**
specifies the receiving field for the data read from the logical unit or terminal, or the application target data area into which data is to be received from the application program connected to the other end of the current conversation.

LENGTH(*data-area***)**
specifies the length, as a halfword binary value, of the data received.

If you specify the INTO option, but omit the MAXLENGTH option, the argument must be a data area that specifies the maximum length that the program accepts. If the value specified is less than zero, zero is assumed.

If you specify the SET option, the argument must be a data area. When the data has been received, the data area is set to the length of the data.

MAXFLENGTH(*data-value***)**
A fullword alternative to MAXLENGTH.

MAXLENGTH(*data-value***)**
specifies the maximum amount (halfword binary value) of data that CICS is to recover. If INTO is specified, MAXLENGTH overrides the use of LENGTH as an input to CICS. If SET is specified, MAXLENGTH provides a way for the program to limit the amount of data it receives at one time.

If the length of data exceeds the value specified and the NOTRUNCATE option is not present, the data is truncated to that value and the LENGERR condition occurs. The data area specified in the LENGTH option is set to the original length of data.

If the length of data exceeds the value specified and the NOTRUNCATE option is present, CICS retains the remaining data and uses it to satisfy subsequent RECEIVE commands. The data area specified in the LENGTH option is set to the length of data returned.

If this option is omitted, the value indicated in the LENGTH option is assumed.

NOTRUNCATE
specifies that, when the data available exceeds the length requested, the remaining data is not to be discarded but is to be retained for retrieval by subsequent RECEIVE commands.

SESSION(*name***)**
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used. If this option is omitted, the principal facility for the task is used.

SET(*ptr-ref***)**
specifies the pointer reference that is to be set to the address of the data read from the logical unit or terminal, or the partner transaction. The pointer reference is valid until the next receive command or the end of task.

If DATALOCATION(ANY) is associated with the application program, the address of the data can be above or below the 16MB line.

If DATALOCATION(BELOW) is associated with the application program, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

STATE(*cvda***)**
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED

- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

Some of the following conditions may occur in combination with others. CICS checks for these conditions in the following order:

1. EODS
2. INBFMH
3. EOC

If more than one occurs, only the first is passed to the application program. EIBRCODE, however, is set to indicate all the conditions that occurred.

06 EOC
occurs when a request/response unit (RU) is received with end-of-chain-indicator set. Field EIBEOC also indicates this condition.

Default action: ignore the condition.

05 EODS (interpreter logical unit only)
occurs when an end-of-data-set indicator is received.

Default action: terminate the task abnormally.

07 INBFMH
occurs if a request/response unit (RU) contains a function management header (FMH). Field EIBFMH contains this indicator and it should be used in preference to INBFMH. The IGNORE CONDITION command can be used to ignore the condition.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

also occurs (RESP2 not set) in any of the following situations:

- The command is used on an APPC conversation that is not using the EXEC CICS interface or that is not a mapped conversation.

Default action: terminate the task abnormally.

22 LENGERR
occurs if data is discarded by CICS because its length exceeds the maximum the program accepts and the NOTRUNCATE option is not specified.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the RECEIVE command is issued by a transaction that has been started as a nonterminal task by the START command, or if the CONVID value or facility specified in the command does not relate to a conversation owned by the application.

Default action: terminate the task abnormally.

24 SIGNAL
occurs when an inbound SIGNAL data-flow control command is received from a partner transaction. EIBSIG is always set when an inbound signal is received.

Default action: ignore the condition.

81 TERMERR
occurs for a session-related or terminal-related error. Any action on that conversation other than a FREE causes an ATCV abend.

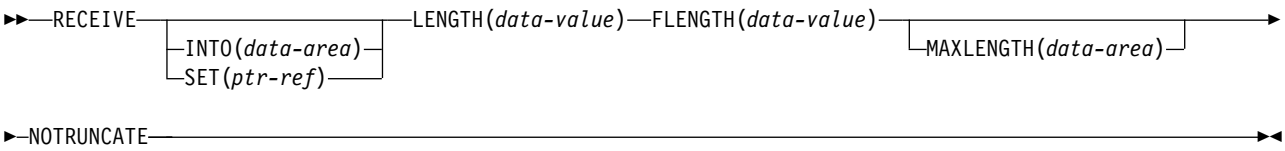
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

RECEIVE (non-z/OS Communications Server default)

Receive data from a task that is not attached to a terminal.

RECEIVE (default)



Conditions: INVREQ, LENGERR, NOTALLOC

Description

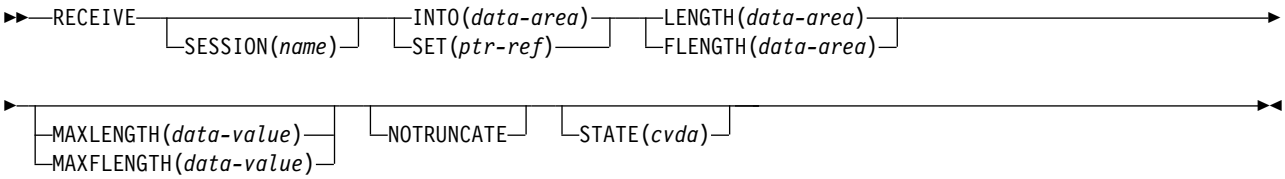
This form of the RECEIVE command is used by all CICS-supported terminals for which the other RECEIVE descriptions are not appropriate.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

RECEIVE (MRO)

Receive data on an MRO conversation.

RECEIVE (MRO)



Conditions: EOC, INBFMH, INVREQ, LENGERR, NOTALLOC, TERMERR

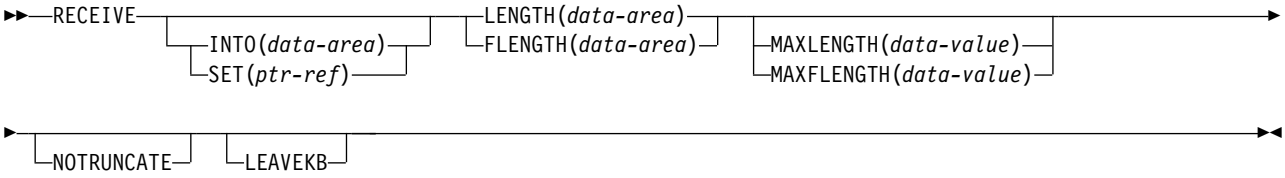
Description

RECEIVE receives data from the conversation partner in an MRO conversation.

RECEIVE (2260)

Receive data from a 2260 or 2265 display station.

RECEIVE (2260)



Conditions: INVREQ, LENGERR

Description

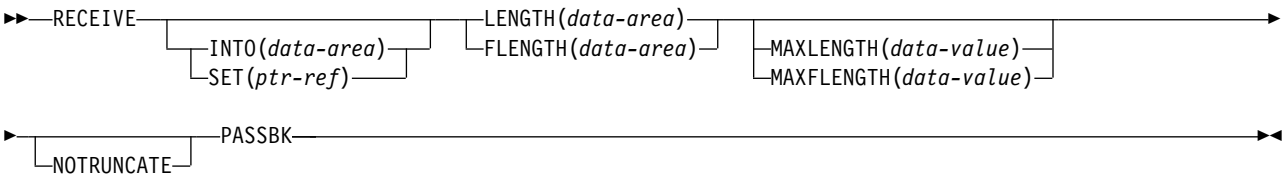
RECEIVE receives data from the terminal.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

RECEIVE (2980)

Receive data from a 2980 general banking terminal system.

RECEIVE (2980)



Conditions: INVREQ, LENGERR, NOPASSBKRD

Description

RECEIVE receives data from the terminal.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) you can omit both the INTO and SET options.

Passbook control

All input and output requests to the passbook area of a 2980 are dependent on the presence of a passbook. The PASSBK option is used to specify that communication is with a passbook. The conditions NOPASSBKRD (RECEIVE) and NOPASSBKWR (SEND) occur on input and output requests respectively when a passbook is not present. These conditions can be handled by a HANDLE CONDITION command and appropriate handling routines.

If the passbook is present on an input request, the application program generally writes back to the passbook area to update the passbook. If the NOPASSBKWR condition occurs, CICS allows immediate output to the terminal. In a routine for the NOPASSBKWR condition, the application program should send an error message to the journal area of the terminal to inform the 2980 operator of this error condition. To allow the operator to insert the required passbook, CICS causes the transaction to wait 23.5 seconds before continuing.

On regaining control from CICS after sending the error message, the application program can attempt again to update the passbook when it has ensured that the print element is positioned correctly in the passbook area. This is generally accomplished by issuing two carrier returns followed by the number of tabs required to move the print element to the correct position.

If the NOPASSBKWR condition occurs during the second attempt to write to the passbook area, the application program can send another error message or take some alternative action (for example, place the terminal out of service). The presence of the Auditor Key on a 2980 Administrative Station Model 2 is controlled by the SEND PASSBK command and may be used in a manner similar to that described above.

Output control

The unit of transmission for a 2980 is called a **segment**. However, for the passbook and journal areas, CICS allows an application program to send messages that exceed the buffer size. For the passbook area, the maximum length of message is limited to one line of a passbook to avoid spacing (*indexing*) past the bottom of the passbook. For the journal area, the maximum length of message is specified in the LENGTH option of the SEND command.

For example, consider a 2972 buffer size of 48 characters and a 2980 Teller Station Model 4 passbook print area of 100 characters per line. The application program can send a message of 100 characters to this area; CICS segments the message to adjust to the buffer size. The application program must insert the passbook indexing character (X'25') as the ***last*** character written in one output request to the passbook area. This is done to control passbook indexing and thereby achieve positive control of passbook presence.

If a message contains embedded passbook indexing characters, and segmentation is necessary because of the length of the message, the output is terminated if the passbook spaces beyond the bottom of the passbook; the remaining segments are not printed.

Output to a common buffer

The SEND CBUFF command is used to transmit data to a common buffer. The data is translated to the character set of the receiving 2980 model. If more than one 2980 model type is connected to the 2972 control unit, the lengths are truncated if they exceed the buffer size.

The DFH2980 structure

The DFH2980 structure contains constants that may only be used when writing COBOL or PL/I application programs for the 2980. The structure is obtained by copying DFH2980 into the application program.

For COBOL, DFH2980 is copied into the working-storage section; for PL/I, DFH2980 is included using a %INCLUDE statement.

The station identification is given in the field STATIONID, whose value must be determined by the ASSIGN command. To test whether a normal or alternate station is being used, the STATIONID field is compared with values predefined in DFH2980. The values are:

```
STATION-n-A or STATION-n-N-

STATION_n_A or STATION_n_N
```

where **n** is an integer (0 through 9) and A and N signify alternate and normal stations. (The break symbol is hyphen (-) for COBOL, and underscore (_) for PL/I.)

The teller identification on a 2980 Teller Station Model 4 is given in the 1-byte character field TELLERID. An ASSIGN command must be used to find out the TELLERID value.

Tab characters (X'05') must be included in the application program. The number of tabs required to position the print element to the first position of a passbook area

is given in the field NUMTAB. An ASSIGN command must be used to find out the NUMTAB value. The value of NUMTAB is specified by the system programmer and may be unique to each terminal.

Other tab characters are inserted as needed to control formatting.

Any of the DFH2980 values TAB-ZERO through TAB-NINE for COBOL and PL/I, may be compared with NUMTAB to find out the number of tab characters that need to be inserted in an output message to get correct positioning of the print element. The tab character is included in DFH2980 as TABCHAR.

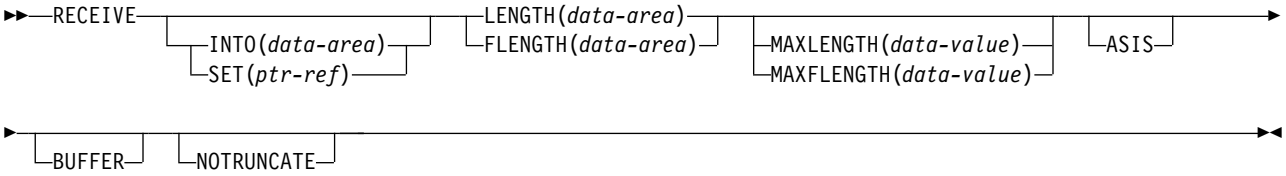
Thirty special characters are defined in DFH2980. Twenty-three of these can be referred to by the name SPECCHAR-# or SPECCHAR_# (for American National Standard COBOL or PL/I) where # is an integer (0 through 22). The seven other characters are defined with names that imply their usage, for example, TABCHAR.

Several other characters defined in DFH2980, such as HOLDPCF or TCTTEPCR, are intended for use in application programs using CICS macros, and should not be required in application programs using CICS commands.

RECEIVE (3790 3270-display)

Receive data from a 3790 (3270-display) logical unit.

RECEIVE (3790 3270-display)



Conditions: INVREQ, LENGERR, TERMERR

Description

RECEIVE receives data from the terminal.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID) (and BUFFER has not been specified), you can omit both the INTO and SET options.

RECEIVE: non-z/OSCommunications Server options

Common options used on the RECEIVE command (non-z/OS Communications Server).

Options

ASIS
indicates that output is to be sent in transparent mode (with no recognition of control characters and accepting any of the 256 possible combinations of eight bits as valid transmittable data).

This option has no effect on the first RECEIVE command of a transaction, because terminal control performs a READ INITIAL operation and uses the terminal defaults to translate the data.

This option has no effect if the screen contains data before a transaction being initiated. This data is read and translated in preparation for the next task and the first RECEIVE command in that task retrieves the translated data.

Note: If you are using a katakana terminal, you might see some messages containing mixed English and katakana characters. That is because katakana terminals cannot display mixed-case output. Uppercase characters in the data stream appear as uppercase English characters, but lowercase characters appear as katakana characters. If this happens, ask your system programmer to specify MSGCASE=UPPER in the system initialization parameters so that messages contain uppercase characters only.

BUFFER
specifies that the contents of the 3270 buffer are to be read, beginning at buffer location one and continuing until all contents of the buffer have been read. All character and attribute sequences (including nulls) appear in the input data stream in the same order that they appear in the 3270 buffer.

FLENGTH(*data-area***)**
A fullword alternative to LENGTH.

INTO(*data-area***)**
specifies the receiving field for the data read from the terminal or logical unit, or the application target area receiving the data from the application program connected to the other end of the current conversation.

If you specify the INTO option, but omit the MAXLENGTH option, the argument for the LENGTH option must be a data area that specifies the maximum length that the program accepts. If the value specified is less than zero, zero is assumed.

If the length of the data exceeds the value specified, but the NOTRUNCATE option is not specified, the data is truncated to that value and the LENGERR condition occurs. When the data has been received, the data area for the LENGTH option is set to the original length of the data.

LEAVEKB
specifies that the keyboard is to remain locked at the completion of the data transfer.

LENGTH(*data-area***)**
specifies the length, as a halfword binary value, of the data transmitted.

If you specify the INTO option, but omit the MAXLENGTH option, the argument must be a data area that specifies the maximum length that the program accepts. If the value specified is less than zero, zero is assumed.

If the length of the data exceeds the value specified, but the NOTRUNCATE option is not specified, the data is truncated to that value and the LENGERR condition occurs. When the data has been received, the data area is set to the original length of the data.

If you specify the SET option, the argument must be a data area. When the data has been received, the data area is set to the length of the data.

For a description of a safe upper limit, see “LENGTH options in CICS commands” on page 10.

MAXFLENGTH(*data-value***)**
A fullword alternative to MAXLENGTH.

MAXLENGTH(*data-value***)**
specifies the maximum amount (halfword binary value) of data that CICS is to recover. If INTO is specified, MAXLENGTH overrides the use of LENGTH as an input to CICS. If SET is specified, MAXLENGTH provides a way for the program to limit the amount of data it receives at one time.

If the value specified is less than zero, zero is assumed.

If the length of data exceeds the value specified and the NOTRUNCATE option is not present, the data is truncated to that value and the LENGERR condition occurs. The data area specified in the LENGTH option is set to the original length of data.

If the length of data exceeds the value specified and the NOTRUNCATE option is present, CICS retains the remaining data and uses it to satisfy subsequent RECEIVE commands. The data area specified in the LENGTH option is set to the length of data returned.

If this option is omitted, the value indicated in the LENGTH option is assumed.

NOTRUNCATE
specifies that, when the data available exceeds the length requested, the remaining data is not to be discarded but is to be retained for retrieval by subsequent RECEIVE commands.

PASSBK
specifies that communication is with a passbook.

PSEUDOBIN
specifies that the data being read is to be translated from System/7 pseudobinary representation to hexadecimal.

SESSION(*name***)**
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used. If this option is omitted, the principal facility for the task is used.

SET(*ptr-ref***)**
specifies a pointer reference that is to be set to the address of data received from the conversation partner in an MRO conversation. The pointer reference is valid until the next receive command or the end of task.

If DATALOCATION(ANY) is associated with the application program, the address of the data can be above or below the 16MB line.

If DATALOCATION(BELOW) is associated with the application program, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

If you specify the SET option, the argument for the LENGTH option must be a data area. When the data has been received, the data area is set to the length of the data.

STATE(*cvda*)
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- FREE
- PENDFREE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

The following conditions can occur in combination with others. CICS checks for these conditions in the order:

1. INBFMH
2. EOC

If more than one occurs, only the first is passed to the application program. EIBRCODE, however, is set to indicate all the conditions that occurred.

08 ENDINPT
occurs when an end-of-input indicator is received.

Default action: terminate the task abnormally.

06 EOC
occurs when a request/response unit (RU) is received with the end-of-chain indicator set. Field EIBEOC also contains this indicator.

Default action: ignore the condition.

04 EOF
occurs when an end-of-file indicator is received.

Default action: terminate the task abnormally.

07 INBFMH
occurs if a request/response unit (RU) contains a function management header (FMH). Field EIBFMH contains this indicator and it should be used in preference to INBFMH. The IGNORE CONDITION command can be used to ignore the condition.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

22 LENGERR
occurs if data is discarded by CICS because its length exceeds the maximum the program accepts and the NOTRUNCATE option is not specified.

Default action: terminate the task abnormally.

50 NOPASSBKRD
occurs if no passbook is present.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

02 RDATT
occurs if a RECEIVE command is terminated by the attention (ATTN) key rather than the return key.

Default action: ignore the condition.

81 TERMERR
occurs for a terminal-related error, such as a session failure. This condition applies to z/OS Communications Server-connected terminals only.

A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

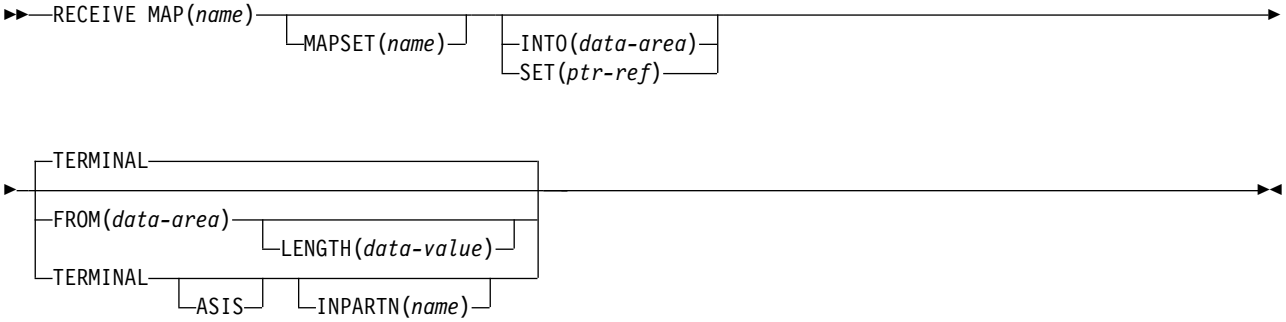
Default action: terminate the task abnormally with abend code ATNI.

RECEIVE MAP

Receive screen input into an application data area.

For further information about BMS, see Basic mapping support .

RECEIVE MAP



Conditions: EOC, EODS, INVMPSZ, INVPARTN, INVREQ, MAPFAIL, PARTNFAIL, RDATT, UNEXPIN

Note: INPARTN is supported by Standard and full BMS

Description

RECEIVE MAP maps input data from a terminal into a data area in an application program.

Data from certain logical units is not mapped, but is left unaltered. Refer to the appropriate CICS subsystem guide to see if this is true for a particular logical unit.

Following a RECEIVE MAP command, the inbound cursor position is placed in EIBCPOSN, and the terminal attention identifier (AID) placed in EIBAID.

See BMS macros for map definitions.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID), you can omit both the INTO and the SET options.

Options

ASIS
specifies that lowercase characters in the 3270 input data stream are not translated to uppercase; this allows the current task to receive a message containing both uppercase and lowercase data.

This option has no effect on the first RECEIVE command of a transaction, or if the screen contains data before a transaction being initiated. For example, if a transaction is initiated by another transaction, and begins by receiving data originally output by that transaction, it cannot suppress uppercase translation on the data. This data is read and translated in preparation for the next task and the first RECEIVE command in that task retrieves the translated data.

Note: If you are using a katakana terminal, you might see some messages containing mixed English and katakana characters. That is because katakana terminals cannot display mixed-case output. Uppercase characters in the data stream appear as uppercase English characters, but lowercase characters appear as katakana characters. If this happens, ask your system programmer to specify MSGCASE=UPPER in the system initialization parameters so that messages contain uppercase characters only.

FROM(*data-area***)**
specifies the data area containing the data to be mapped by a RECEIVE MAP command. This includes the 12-byte prefix generated by the TIOAPFX=YES option on the DFHMDI and DFHMSD BMS map definitions (see DFHMDI operands, TERM and specifying NODDS in the BMS operand).

INPARTN(*name***)**
specifies the name (1–2 characters) of the partition in which the terminal operator is expected to enter data. If the terminal operator enters data in some other partition, the INPARTN partition is activated, the keyboard is unlocked for the partition, and an error message is output to any error message partition. This option is ignored if the terminal does not support partitions, or if there is no application partition set.

INTO(*data-area***)**
specifies the data area into which the mapped data is to be written. If this field is not specified, the name defaults to the name of the map suffixed with an I.

LENGTH(*data-value***)**
specifies the length of the data to be formatted as a halfword binary value. It must not exceed the length of the FROM data area, but this should include the length of the 12-byte prefix generated by the TIOAPFX=YES option on the DFHMDI and DFHMSD BMS map definitions (see DFHMDI operands, TERM and specifying NODDS in the BMS operand).

For a description of a safe upper limit, see “LENGTH options in CICS commands” on page 10.

MAP(*name***)**
specifies the name (1–7 characters) of the map to be used.

MAPSET(*name***)**
specifies the unsuffixed name (1–7 characters) of the mapset to be used. The mapset must reside in the CICS program library. The mapset can be defined either by using RDO or by program autoinstall when the mapset is first used. If this option is not specified, the name given in the MAP option is assumed to be that of the mapset.

SET(*ptr-ref***)**
specifies the pointer that is to be set to the address of the 12-byte prefix to the mapped data. The 12-byte prefix is generated by the TIOAPFX=YES option on the DFHMDI and DFHMSD BMS map definitions (see DFHMDI operands, TERM and specifying NODDS in the BMS operand).

The pointer reference is valid until the next receive command or the end of task.

If DATALOCATION(ANY) is associated with the application program, the address of the data may be above or below the 16MB line. The data area includes the 12-byte prefix. Ensure that the 12-byte prefix is not overwritten, because this might cause a storage violation.

If DATALOCATION(BELOW) is associated with the application program, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

TERMINAL
specifies that input data is to be read from the terminal that originated the transaction.

Conditions

Some of the following conditions can occur in combination. If more than one occurs, only the first is passed to the application program.

EIBRCODE, however, is set to indicate all the conditions that occurred.

06 EOC
occurs if the request/response unit (RU) is received with the end-of-chain (EOC) indicator set. It applies only to logical units.

Default action: ignore the condition.

05 EODS
occurs if no data is received (only an FMH). It applies only to 3770 batch LUs and to 3770 and 3790 batch data interchange LUs.

Default action: terminate the task abnormally.

38 INVMPSZ
occurs if the specified map is too wide or too long for the terminal.

Default action: terminate the task abnormally.

65 INVPARTN
occurs if the specified partition is not defined in the partition set associated with the application program.

Default action: terminate the task abnormally.

16 INVREQ
occurs if a RECEIVE MAP command is issued in a nonterminal task; these tasks do not have a TIOA or a TCTTE.

Default action: terminate the task abnormally.

36 MAPFAIL
occurs if the data to be mapped has a length of zero or does not contain a set-buffer-address (SBA) sequence. It applies only to 3270 devices. The receiving data area contains the unmapped input data stream. The amount of unmapped data moved to the user's area is limited to the length specified in the LENGTH option. The input map is not set to nulls.

This condition also arises if a program issues a RECEIVE MAP command to which the terminal operator responds by pressing a CLEAR or PA key, or by pressing ENTER or a function key without entering data.

Default action: terminate the task abnormally.

66 PARTNFAIL
occurs if the terminal operator attempts to enter data more than three times in a partition other than that specified by the INPARTN option.

Default action: terminate the task abnormally.

02 RDATT
occurs if a RECEIVE MAP command is terminated by the operator using the ATTN key rather than the RETURN key. It applies only to the 2741 Communications Terminal, and only if 2741 read attention support has been generated for CICS.

Default action: ignore the condition.

49 UNEXPIN
occurs when unexpected or unrecognized data is received. This only applies to batch data interchange terminals.

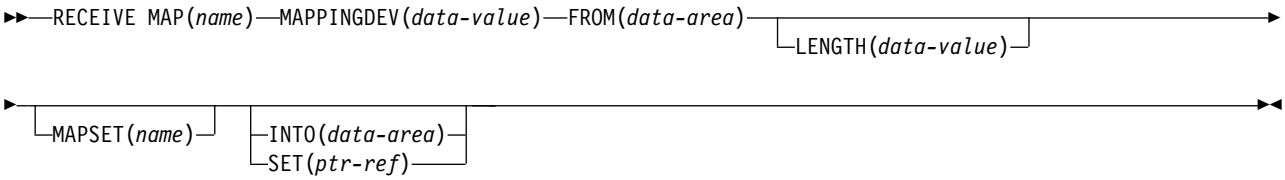
Default action: terminate the task abnormally.

RECEIVE MAP MAPPINGDEV

Receive screen input into an application data area, without reference to the principal facility, if any. Terminal characteristics are obtained from the **MAPPINGDEV** parameter.

For further information about BMS, see Basic mapping support.

RECEIVE MAP MAPPINGDEV



Conditions: INVMPSZ, INVREQ, MAPFAIL,

Description

RECEIVE MAP MAPPINGDEV allows the mapping of input data from a 3270 terminal that is not necessarily the principal facility of the transaction.

MAPPINGDEV specifies the name of a 3270 terminal whose BMS characteristics were used to create the input data stream. This might be a terminal from which the data was originally received using a RECEIVE command.

Options

AID(*data-value***)**
Specifies the one-byte data area containing the value of the 3270 attention identifier (AID) to be used when performing the mapping operation. Usually this will be the value contained in EIBAID following the RECEIVE operation that originally received the datastream from the terminal.

The value specified is moved into field EIBAID in the EXEC interface block on completion of the operation. No check is made that the AID value specified is valid.

If AID(data-value) is not specified, then the AID value defaults to X'7D' (the Enter key).

If the AID byte (either explicitly, or by default) indicates an operation other than CLEAR, PA1, PA2, or PA3, and CURSLOC=YES is specified for the map, then the field containing the cursor is flagged by setting the X'02' bit in its flag byte.

If the AID (whether specified explicitly, or by default) is the subject of a HANDLE AID command, the specified branch will be taken in the usual way.

CURSOR(*data-value***)**
Specifies an unsigned halfword binary field containing the cursor position (relative to zero) to be used. Usually this will be the value contained in EIBCPOSN following the RECEIVE operation that originally received the datastream from the terminal.

The value specified is moved into EIBCPOSN in the EXEC interface block on completion of the operation. No check is made that the CURSOR value specified is valid.

If CURSOR(data-value) is not specified, then the cursor value defaults to X'0000'.

FROM(*data-area***)**
Specifies the data area containing the data to be mapped. This must be in the format of a TIOA and must contain a 12-byte prefix.

INTO(*data-area***)**
Specifies the data area into which the mapped data is to be written. If this field is not specified, the name defaults to the name of the map suffixed with an I.

LENGTH(*data-value***)**
Specifies the length of the data to be formatted as a halfword binary value. It must not exceed the length of the FROM data area, but this should include the length of the 12-byte prefix generated by the TIOAPFX=YES option on the DFHMDI and DFHMSD BMS map definitions (see DFHMDI operands, TERM and specifying NODDS in the BMS operand). For a description of a safe upper limit, see “LENGTH options in CICS commands” on page 10.

MAP(*name***)**
Specifies the name (1–7 characters) of the map to be used.

MAPPINGDEV(*data-value***)**
Specifies the name of a 3270 terminal whose characteristics match those of the terminal from which the data was originally received using a RECEIVE command.

MAPSET(*name***)**
Specifies the unsuffixed name (1–7 characters) of the mapset to be used. The mapset must reside in the CICS program library. The mapset can be defined either by using RDO or by program autoinstall when the mapset is first used. If this option is not specified, the name given in the MAP option is assumed to be that of the mapset.

SET(*ptr-ref***)**
Specifies the pointer that is to be set to the address of the 12-byte prefix to the mapped data. The pointer reference is valid until the next RECEIVE or RECEIVE MAP command, or until the end of the transaction, unless the application releases it by using a FREEMAIN request.

If TASKDATALOC(ANY) is specified for the running task, the data returned may be above or below the 16MB line.

If TASKDATALOC(BELOW) is specified for the running task,the data returned is below the 16MB line.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in CICS-key.

Conditions

Some of the following conditions may occur in combination. If more than one occurs, only the first is passed to the application program.

38 INVMPSZ
occurs if the specified map is too wide or too long for the terminal.

Default action: terminate the task abnormally.

16 INVREQ
occurs if the terminal specified by MAPPINGDEV does not exist, does not support BMS, or is not a 3270 printer or display.

Default action: terminate the task abnormally.

36 MAPFAIL
occurs if the data to be mapped has a length of zero or does not contain a set-buffer-address (SBA) sequence.

Default action: terminate the task abnormally.

RECEIVE PARTN

Receive data from an 8775 terminal partition. This command is only available on standard and full BMS.

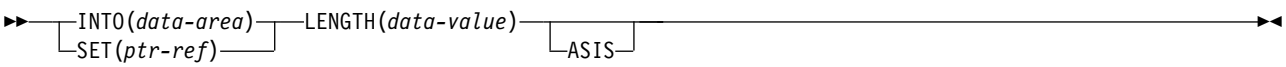
For further information about BMS, see Basic mapping support.

RECEIVE PARTN



Standard and full BMS:

RECEIVE PARTN



Conditions: EOC, EODS, INVPARTN, INVREQ, LENGERR

Description

RECEIVE PARTN reads data from a partition on an 8775 terminal. It indicates which partition the data came from, and puts the data into the INTO or the SET data area. You can then treat the data as though it had originated from a terminal in base (unpartitioned) state.

Following a RECEIVE PARTN command, the inbound cursor position is placed in EIBCPOSN, and the terminal attention identifier (AID) placed in EIBAID. EIBAID and EIBCPOSN are also updated at task initiation for non-ATI tasks as well as after each terminal control and BMS input.

See BMS macros for map definitions.

If data is to be received, you must specify either the INTO or the SET option. If a RECEIVE is issued purely to detect an attention identifier (AID), you can omit both the INTO and the SET options.

Options

ASIS
specifies that lowercase characters in the 3270 input data stream are not translated to uppercase; this allows the current task to receive a message containing both uppercase and lowercase data.

The ASIS option has no effect on the first RECEIVE command of a transaction, or if the screen contains data prior to a transaction being initiated. For example, if a transaction is initiated by another transaction, and begins by receiving data originally output by that transaction, it cannot suppress uppercase translation on the data. This data is read and translated in preparation for the next task and the first RECEIVE command in that task retrieves the translated data.

Note: If you are using a katakana terminal, you might see some messages containing mixed English and katakana characters. That is because katakana terminals cannot display mixed-case output. Uppercase characters in the data stream appear as uppercase English characters, but lowercase characters appear as katakana characters. If this happens, ask your system programmer to specify MSGCASE=UPPER in the system initialization parameters so that messages contain uppercase characters only.

INTO(*data-area*)
specifies the area into which the input data stripped of partition controls is to

be written. The length of this area must be specified by the LENGTH option. If the area is not large enough to hold the input data, the input data is truncated, and the LENGERR condition raised. The length option data area is set to the length of data received, before any truncation.

LENGTH(*data-value***)**
specifies the length of the data to be formatted as a halfword binary value. It must be set to the length of any INTO area before the command. After the command, BMS sets the LENGTH option to the length of data received before any truncation if the INTO area is too small.

For a description of a safe upper limit, see "LENGTH options in CICS commands" on page 10.

PARTN(*data-area***)**
is set to the name (1–2 characters) of the input partition. The partition can be defined either by using RDO or by program autoinstall when the partition is first used.

SET(*ptr-ref***)**
specifies the pointer that is to be set to the address of the 12-byte prefix to the mapped data. The pointer reference is valid until the next receive command or the end of task.

If DATALOCATION(ANY) is associated with the application program, the address of the data may be above or below the 16MB line.

If DATALOCATION(BELOW) is associated with the application program, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

Conditions

Some of the following conditions can occur in combination. If more than one occurs, only the first one is passed to the application program.

06 EOC
occurs if the request/response unit (RU) is received with the end-of-chain (EOC) indicator set. It applies only to logical units.

Default action: ignore the condition.

05 EODS
occurs if no data is received (only an FMH). It applies only to 3770 batch LUs and to 3770 and 3790 batch data interchange LUs.

Default action: terminate the task abnormally.

65 INVPARTN
occurs if the specified partition is not defined in the partition set associated with the application program.

Default action: terminate the task abnormally.

16 INVREQ
occurs if a RECEIVE PARTN command is issued in a nonterminal task; these tasks do not have a TIOA or a TCTTE.

Default action: terminate the task abnormally.

22 LENGERR
occurs if the INTO area of a RECEIVE PARTN command is not large enough to hold the input data.

Default action: truncate the data to fit within the INTO area.

RELEASE

Release a loaded program, table, or mapset.

RELEASE



Conditions: INVREQ, NOTAUTH, PGMIDERR

This command is threadsafe.

Note for dynamic transaction routing: Using RELEASE of a program LOADed with HOLD could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. See Affinity for more information about transaction affinities.

Description

RELEASE releases the program, table, or mapset previously loaded by a LOAD command. This means that the issuing task can no longer use the resource unless another LOAD is issued.

Note: RELEASE does not remove a program from storage. It reduces the RESCOUNT by 1 and when the count reaches zero, the storage occupied by the program can be released by CICS storage manager.

If the HOLD option is specified in the LOAD command, the loaded resource is not released at the end of the task. It can only be released by a RELEASE command. This RELEASE command may be issued by the task that loaded the resource or by any other task.

If the HOLD option is not specified in the LOAD command, the loaded resource is released at the end of the task. It may, however, be released before this by the task that loaded the resource issuing a RELEASE command.

Options

PROGRAM(*name*)
specifies the identifier (1–8 characters) of a program, table, or mapset to be released.

Conditions

16 INVREQ
RESP2 values:

5 An invalid attempt is made by the program to release itself. A RELEASE command for the program that contains this command is allowed only when a corresponding LOAD command for the program

has been issued from the same task, or when a LOAD command with the HOLD option has been issued from another task.

6 The command is issued for a program that is not loaded.

7 Either the command is issued for a program that was loaded, without the HOLD option, by another task; or the program has been enabled as a global user exit .

17 The program is defined with RELOAD=YES. It must be released by a FREEMAIN rather than a RELEASE command.

30 The program manager domain has not yet been initialized. This is probably due to a release request having been made in a first stage PLT.

Default action: terminate the task abnormally.

70 NOTAUTH
occurs when a resource security check has failed on PROGRAM(name).

Default action: terminate the task abnormally.

27 PGMIDERR
RESP2 values:

1 A program, table, or mapset has no installed resource definition.

2 A program, table, or mapset is disabled.

9 The installed program definition is for a remote program.

42 An attempt has been made to RELEASE a JVM program. This is invalid because Java byte codes programs are not managed by CICS Loader.

Default action: terminate the task abnormally.

Examples

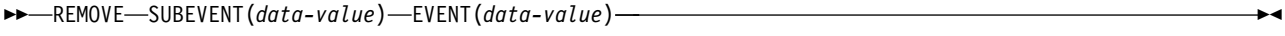
The following example shows how to release an application program, called PROG4, loaded in response to a LOAD command:

```
EXEC CICS RELEASE PROGRAM('PROG4')
```

REMOVE SUBEVENT

Remove a sub-event from a BTS composite event.

REMOVE SUBEVENT



Conditions: EVENTERR, INVREQ

Description

REMOVE SUBEVENT removes a sub-event from a named BTS composite event.

This call does not delete the removed event. Nor does it reset the event's fire status. Note that, after this call, the removed event—because it is no longer a sub-event—will cause the current activity to be reattached if it fires.

Removing a sub-event causes the composite's predicate to be re-evaluated.

Options

EVENT(data-value)
specifies the name (1–16 characters) of the composite event.

SUBEVENT(data-value)
specifies the name (1–16 characters) of the event which is to be removed from the named composite event.

Conditions

111 EVENTERR
RESP2 values:

4 The event specified on the EVENT option is not recognized by BTS.

5 The sub-event specified on the SUBEVENT option is not recognized by BTS.

16 INVREQ
RESP2 values:

1 The command was issued outside the scope of an activity.

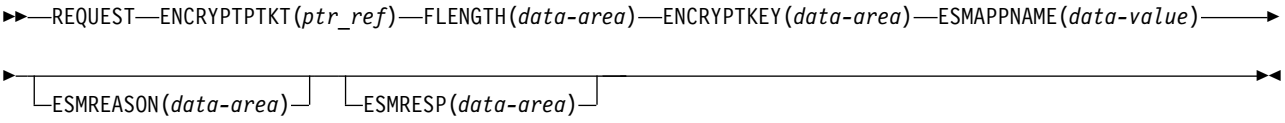
2 The event specified on the EVENT option is not a composite event.

3 The event specified on the SUBEVENT option is not a sub-event of the composite event specified on the EVENT option.

REQUEST ENCRYPTPTKT

Request an external security manager to generate an encrypted PassTicket.

REQUEST ENCRYPTPTKT



Conditions: NOTAUTH, INVREQ

This command is threadsafe.

Description

The **REQUEST ENCRYPTPTKT** command requests an external security manager (ESM), such as RACF, to generate an encrypted PassTicket.

A PassTicket that is not encrypted is a password substitute that your program can use to sign on to a particular application on a particular system, such as another

CICS region. You must use the ESMAPPNAME option to specify the profile name by which the external security manager refers to the application to which you are signing on.

ENCRYPTKEY must contain a value that is returned by a previous **VERIFY TOKEN** command for the same CICS task.

Note: When a **VERIFY TOKEN** command is run with the ENCRYPTKEY option, any previous value of ENCRYPTKEY that was obtained for the same task is no longer valid. It means if a number of **VERIFY TOKEN** commands are run, it is only an ENCRYPTKEY value from the last one that is valid to use with a **REQUEST ENCRYPTPTKT** command.

A value in ENCRYPTKEY can be used once to obtain an encrypted PassTicket with a **REQUEST ENCRYPTPTKT** command. Having run a **REQUEST ENCRYPTPTKT** command once, if another encrypted PassTicket is required it is necessary to run a **VERIFY TOKEN** command again to obtain a new value with the ENCRYPTKEY option. Then, you can run a **REQUEST ENCRYPTPTKT** command for a second time.

The encrypted PassTicket returned by a **REQUEST ENCRYPTPTKT** command is intended for decryption by the subsystem that supplied the Kerberos token, and that was used with the **VERIFY TOKEN** command with ENCRYPTKEY option.

The encrypted PassTicket that the external security manager generates is for the user ID associated with the task that issues the **REQUEST ENCRYPTPTKT** command. Use the **EXEC CICS ASSIGN** command with the USERID option to identify the user ID that is associated with the task. A PassTicket cannot be obtained for the default user ID.

Note: A request for a PassTicket succeeds if a userid is revoked, but an attempt to sign on with that userid and PassTicket fails.

A PassTicket must be used within 10 minutes of being generated. If the PassTicket times out (because, for example, of a session failure), your application must generate another PassTicket before you can attempt to sign on again. Repeated failed sign-on attempts with PassTickets can result in the user ID being revoked.

Before you use PassTickets, you must ensure that the system clocks for the target system and the originating system are synchronized to within the valid time range. You must also define a Secure Signon key for each target system. For information on the requirements for using PassTickets, see security.racf.passticket.

Options

ENCRYPTPTKT(*ptr-ref***)**
Specifies a data area in which the encrypted PassTicket is returned.

FLENGTH(*fullword binary data-area***)**
Returns the length of the encrypted PassTicket.

ENCRYPTKEY(*data-area***)**
This is the 4-byte token that is obtained from a previous VERIFY TOKEN representing the key to be used to encrypt the PassTicket.

ESMAPPNAME(*data-value***)**
Specifies the eight-character profile name by which the external security manager refers to the application for which the supplied PassTicket is used. For example, if the application is another CICS region, the profile name might

be the APPLID of the CICS region. If the external security manager is RACF, for more information about RACF profile names and PassTickets, see Using the secured signon function in z/OS Security Server RACF Security Administrator's Guide.

ESMRESP(*data-area***)**
Returns the response code from the external security manager, as a fullword binary variable. When RACF is in use, the possible values are the return codes that are supplied when you are using the RACF secured signon service to generate a PassTicket, as follows:

0 A PassTicket was produced.

8 A PassTicket was not produced.

ESMREASON(*data-area***)**
Returns the reason code from the external security manager, as a fullword binary variable.

If the external security manager is RACF, the possible values are the RACF reason codes that are supplied by the **z/OS Security Server RACF Callable Services** > **GSS-API functions (Function code 2)** > **Subfunction codes** > **Wrap a message**. SeeSubfunction codes in z/OS Security Server RACF Callable Services. For an explanation of a reason code, see R_GenSec Return and reason codes in z/OS Security Server RACF Callable Services and Status codes in z/OS Integrated Security Services Network Authentication Service Administration.

Conditions

16 INVREQ
RESP2 values:

251 The interface between CICS and the external security manager is not active.

252 The value that is returned by the external security manager in ESMRESP is not classified by CICS.

254 The external security manager does not support requests for a PassTicket.

255 Invalid ENCRYPTKEY.

256 This command is not valid when you are running under the default user ID.

257 The ENCRYPTTOKEN option was specified, but the associated kerberos token originated from a system that does not support message confidentiality.

70 NOTAUTH
RESP2 values:

250 PassTicket not built successfully. The external security manager does not authorize a request for a PassTicket for the combination of the user ID associated with the task that issued this command, and the profile name that is specified in ESMAPPNAME.

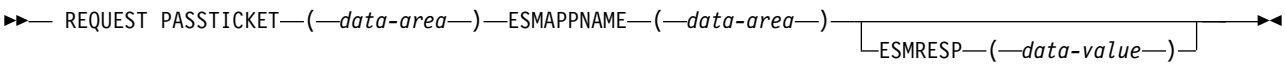
260 The external security manager does not authorize a request to generate a PassTicket for this region.

REQUEST PASSTICKET

Request an external security manager to build a PassTicket.

Syntax

REQUEST PASSTICKET



Conditions: NOTAUTH, INVREQ

This command is threadsafe.

Description

The **REQUEST PASSTICKET** command requests an external security manager (ESM), such as RACF, to build a PassTicket. The PassTicket is a password substitute that your program can use to sign on to a particular application on a particular system, such as another CICS region. You must use the ESMAPPNAME option to specify the profile name by which the external security manager refers to the application to which you are signing on.

The PassTicket that the external security manager generates is for the user ID associated with the task that issues the **REQUEST PASSTICKET** command. Use the **EXEC CICS ASSIGN** command with the USERID option to identify the user ID that is associated with the task.

Note: A request for a PassTicket succeeds if a userid is revoked, but an attempt to sign on with that userid and PassTicket fails.

A PassTicket must be used within 10 minutes of being generated. If the PassTicket times out (because, for example, of a session failure), your application must generate another before attempting to sign on again. Repeated failed sign-on attempts with PassTickets can result in the user ID being revoked.

PassTickets are not displayed when the CICS execution diagnostic facility (EDF) is used.

Before using PassTickets, you must ensure that the system clocks for the target system and the originating system are synchronized to within the valid time range. You must also define a Secure Signon key for each target system. For information on the requirements for using PassTickets, see Generating and using PassTickets for secure sign-on.

Options

ESMAPPNAME(*data-area***)**
Specifies the eight-character profile name by which the external security manager refers to the application for which the supplied PassTicket is used. For CICS regions, the profile name is the APPLID of the CICS region. If the external security manager is RACF, for more information about RACF profile names and PassTickets, see Using the secured signon function in z/OS Security Server RACF Security Administrator's Guide.

ESMRESP(*data-area*)
Returns the response code from the external security manager, as a fullword binary variable. When RACF is in use, the possible values are the return codes that are supplied when using the RACF secured signon service to generate a PassTicket, as follows:

0 A PassTicket was produced.

8 A PassTicket was not produced.

PASSTICKET(*data-area*)
Returns the 8-character PassTicket generated by the external security manager.

Conditions

16 INVREQ
RESP2 values:

247 An invalid value has been specified for ESMAPPNAME.

251 The interface between CICS and the external security manager is not active.

252 The value returned by the external security manager in ESMRESP is not classified by CICS.

254 The external security manager does not support requests for a PassTicket.

256 This command is not valid when running under the default userid.

70 NOTAUTH
RESP2 values:

250 PassTicket was not built successfully. The external security manager does not authorize a request for a PassTicket for the combination of the user ID associated with the task that issued this command, and the profile name specified in ESMAPPNAME.

260 The external security manager does not authorize a request to generate a PassTicket for this region.

RESET ACQPROCESS

Reset a BTS process to its initial state.

RESET ACQPROCESS



Conditions: INVREQ, IOERR, LOCKED, NOTAUTH, PROCESSBUSY, PROCESSERR

Description

RESET ACQPROCESS resets the currently-acquired BTS process to its initial state. Any descendant activities of the root activity are deleted.

Note: RESET has no effect on the process containers, nor on the root activity's containers, the contents of which are unchanged.

Issue this command, before a second RUN command, when a process needs to be retried. When the process is re-run, the root activity is sent a DFHINITIAL event.

To be eligible to be reset, a process must:

1. Have been acquired in the current unit of work—that is, it must be the currently-acquired process.
2. Be in one of the following modes:
   - COMPLETE. This is the usual case. Perhaps the process has completed abnormally, and needs to be reset before being retried.
   - INITIAL. The process has not yet been run.

Options

ACQPROCESS
specifies that the process that is currently acquired by the requestor is to be reset.

Conditions

16 INVREQ
RESP2 values:

15 The unit of work that issued the request has not acquired a process.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to reset the process.

106 PROCESSBUSY
RESP2 values:

13 The request timed out. It may be that another task using this process-record has been prevented from ending.

108 PROCESSERR
RESP2 values:

14 The process to be reset is not in COMPLETE or INITIAL mode.

RESET ACTIVITY

Reset a BTS activity to its initial state.

RESET ACTIVITY



Conditions: ACTIVITYBUSY, ACTIVITYERR, INVREQ, IOERR, LOCKED, NOTAUTH

Description

RESET ACTIVITY resets a BTS child activity to its initial state. Its completion event is added to the parent's event pool, with the fired status set to NOTFIRED. If the activity has children of its own, they are deleted.

Note: RESET has no effect on the contents of the activity's data containers, which are unchanged.

Issue this command, before a second RUN command, when an activity needs to be retried. When the activity is re-run, it is sent a DFHINITIAL event.

To be eligible to be reset, an activity must:

1. Be a child of the activity that issues the RESET command.
2. Be in one of the following modes:
   - COMPLETE. This is the usual case. Perhaps the activity has completed abnormally, and needs to be reset before being retried.
   - INITIAL. The activity has not yet been run.

Options

ACTIVITY(data-value)
specifies the name (1–16 characters) of the activity to be reset. This must be a child of the current activity.

Conditions

107 ACTIVITYBUSY
RESP2 values:

19 The request timed out. It may be that another task using this activity-record has been prevented from ending.

109 ACTIVITYERR
RESP2 values:

8 The activity named in the ACTIVITY option is not a child of the current activity.

14 The activity to be reset is not in COMPLETE or INITIAL mode.

16 INVREQ
RESP2 values:

4 The RESET ACTIVITY command was issued outside the scope of a currently-active activity.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

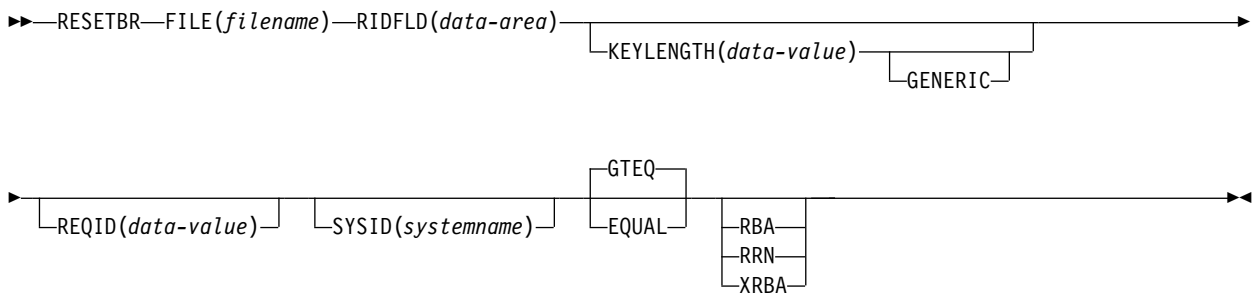
70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to reset the activity.

RESETBR

Reset the start of a browse.

RESETBR



Conditions: FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, NOTAUTH, NOTFND, SYSIDERR

This command is threadsafe if the file to which it refers is:
- Defined as remote and the command is function shipped over an IPIC connection to a remote CICS region.
- Defined as either local VSAM or RLS.

This command is not threadsafe if the file to which it refers is:
- Defined as remote and the command is function shipped over a non-IPIC connection.
- Defined as a shared data table, coupling facility data table, or BDAM file.

Description

RESETBR specifies, during a browse, the record in a file or data table on a local or a remote system, where you want the browse to be repositioned.

When browsing a VSAM file or data table, you can use the RESETBR command to reposition the browse (which you can also achieve by modifying the RIDFLD data area on a READNEXT or READPREV command) and you can also change the

browse characteristics from those specified on STARTBR, without ending the browse. The characteristics that can be changed are those specified by the GENERIC, GTEQ, and RBA options.

When browsing a BDAM file, you can include this command at any time before issuing any other browse command. It is similar to an ENDBR–STARTBR sequence (but with less function), and gives the BDAM user the sort of skip sequential capability that is available to VSAM users through use of the READNEXT command.

If a RESETBR request specifies the precise key at which the browse is to start (that is, it specifies a full key and the EQUAL keyword) the record returned on the following READNEXT (or READPREV) command might not be the same as the record specified by the RESETBR command for a file opened in VSAM NSR or RLS mode. This can occur because the initial record specified on the RESETBR command can be deleted by another transaction in between the RESETBR completing and a READNEXT or READPREV command being issued. In VSAM LSR mode, the initial record cannot be deleted between the RESETBR and the READNEXT.

Note: RESETBR invalidates a TOKEN set by a previous READ or READNEXT command.

Options

EQUAL
specifies that the search is satisfied only by a record having the same key (complete or generic) as that specified in the RIDFLD option.

FILE(*filename*)
(VSAM and data table) specifies the name of the file to be accessed.

If SYSID is specified, the data set to which this file refers is assumed to be on a remote system irrespective of whether the name is defined to CICS. Otherwise, the resource definition is used to find out whether the data set is on a local or a remote system.

GENERIC
(VSAM KSDS, path or data table) specifies that the search key is a generic key whose length is specified in the KEYLENGTH option. The search for a record is satisfied when a record is found that has the same starting characters (generic key) as those specified.

GTEQ
(VSAM and data table) specifies that if the search for a record that has the same key (complete or generic) as that specified in the RIDFLD option is unsuccessful, the first record that has a greater key is retrieved. Use this option only with keyed or RRN.

KEYLENGTH(*data-value*)
specifies the length (halfword binary) of the key that has been specified in the RIDFLD option, except when RBA or RRN is specified, in which case KEYLENGTH is not valid.

This option must be specified if GENERIC is specified, and it can be specified whenever a key is specified. If the length specified is different from the length defined for the data set and the operation is not generic, the INVREQ condition occurs.

The INVREQ condition also occurs if a RESETBR command specifies GENERIC, and the KEYLENGTH value is not less than that specified in the VSAM definition.

If KEYLENGTH(0) is used with the object of reading the first record in the data set, the GTEQ option must also be specified. If EQUAL is specified either explicitly or by default with KEYLENGTH(0), the results of the STARTBR are unpredictable.

For remote files, the KEYLENGTH can be specified in the FILE definition. If KEYLENGTH is not defined there, and is not specified in the application program, and the key is longer than 4 characters, the default value is 4.

RBA
(VSAM KSDS or ESDS base data sets, or CICS-maintained data tables only, not paths) specifies that the record identification field specified in the RIDFLD option contains a relative byte address. Use this option only when browsing an ESDS or KSDS base and using relative byte addresses instead of keys to identify the records.

You cannot use RBA for:

- User-maintained data tables
- Coupling facility data tables
- Any KSDS file opened in RLS access mode
- KSDS files that use extended addressing

Also, you are recommended not to use RBA for ESDS files that hold more than 4 GB. (Use XRBA instead.)

REQID(*data-value***)**
specifies as a halfword binary value a unique request identifier for a browse, used to control multiple browse operations on a data set. If this option is not specified, a default value of zero is assumed.

RIDFLD(*data-area***)**
specifies the record identification field. The contents can be a key, a relative byte address, or a relative record number (for VSAM data sets), or a block reference, physical key, and a deblocking argument (for BDAM data sets). For a relative byte address or a relative record number, the format of this field must be fullword binary. For a relative byte address, the RIDFLD value can be greater than or equal to zero. For a relative record number, the RIDFLD value can be greater than or equal to 1.

For VSAM, a full record ID of X'FF's indicates that the browse is to be positioned at the end of the data set in preparation for a backwards browse using READPREV commands.

RRN
(VSAM RRDS) specifies that the record identification field specified in the RIDFLD option contains a relative record number.

SYSID(*systemname***)**
specifies the name of the system to which the request is directed.

If you specify SYSID, and omit both RBA and RRN, you must also specify KEYLENGTH; it cannot be found in the resource definition.

XRBA
specifies that the record identification field specified in the RIDFLD option contains an extended relative byte address. This option should be used when browsing records in an ESDS extended addressing data set.

You cannot specify XRBA on a RESETBR command unless the associated STARTBR command also specified XRBA.

Conditions

12 FILENOTFOUND
RESP2 values:

1 A file name referred to in the FILE option is not defined to CICS.

Default action: terminate the task abnormally.

21 ILLOGIC
RESP2 values VSAM):

110 A VSAM error occurs that is not in one of the other CICS response categories.

See EIBRCODE in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

25 The KEYLENGTH and GENERIC options are specified, and the length specified in the KEYLENGTH option is greater than or equal to the length of a full key.

26 The KEYLENGTH option is specified (but the GENERIC option is not specified), and the specified length does not equal the length defined for the data set to which this file refers.

36 The REQID, SYSID, or file name does not match that of any successful STARTBR command.

37 The type of record identification (for example, key or relative byte address) used to access a data set during the browse has been changed. You cannot specify one type of addressing on STARTBR and another on RESETBR.

42 The KEYLENGTH and GENERIC options are specified, and the length specified in the KEYLENGTH option is less than zero.

44 The command does not conform to the format of RESETBR for a user-maintained or coupling facility data table; for example, RBA is specified.

51 A RESETBR command to a KSDS file that is being accessed in RLS mode specifies the RBA keyword. RLS mode does not support RBA access to KSDS data sets.

Default action: terminate the task abnormally.

17 IOERR
RESP2 values:

120 There is an I/O error during the file control operation. An I/O error is any unusual event that is not covered by a CICS condition.

For VSAM files, IOERR usually indicates a hardware error. Further information is available in the EXEC interface block; for details, see EIB fields.

For a coupling facility data table, an IOERR indicates a bad response returned from a coupling facility access.

Default action: terminate the task abnormally.

54 ISCINVREQ
RESP2 values:

70 The remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

70 NOTAUTH
RESP2 values:

101 A resource security check has failed on FILE(*filename*).

Default action: terminate the task abnormally.

13 NOTFND
RESP2 values:

80 An attempt to retrieve a record based on the search argument provided is unsuccessful.

NOTFND can also occur if a generic RESETBR with KEYLENGTH(0) specifies the EQUAL option.

81 XRBA was specified, and the value of RIDFLD was greater than 4 GB, but the data set is not an extended addressing ESDS.

Default action: terminate the task abnormally.

53 SYSIDERR
RESP2 values:

130 The SYSID option specifies a name that is neither the local CICS region nor a remote system (as defined by a CONNECTION definition). SYSIDERR also occurs when the link to the remote system is closed

131 For a coupling facility data table, the connection to the coupling facility data table server has failed. This could be because the server itself has failed, or the server is available, but CICS has failed to connect to it.

132 The RESETBR is issued against a coupling facility data table that no longer exists, probably because of a coupling facility failure, in which case the coupling facility data table server also fails. See Setting up and running a coupling facility data table server for information about restarting a coupling facility data table server and reloading a table.

Default action: terminate the task abnormally.

RESUME

Resume a suspended BTS process or activity.

RESUME



Conditions: ACTIVITYBUSY, ACTIVITYERR, INVREQ, IOERR, LOCKED, PROCESSERR

Description

RESUME resumes a BTS process or activity that has previously been suspended (by means of a SUSPEND command). That is, it allows the process or activity to be reattached when events in its event pool are fired. If events that would normally have caused reattachment have occurred during the time the process or activity was suspended, the latter is reattached for all these events.

The only process a program can resume is the one it has acquired in the current unit of work.

The only activities a program can resume are as follows:

- If it is running as the activation of an activity, its own child activities. It can resume several of its child activities within the same unit of work.
- The activity it has acquired, by means of an ACQUIRE ACTIVITYID command, in the current unit of work.

Options

ACQACTIVITY
specifies that the activity to be resumed is the one that the current unit of work has acquired by means of an ACQUIRE ACTIVITYID command.

ACQPROCESS
specifies that the process that is currently acquired by the requestor is to be resumed.

ACTIVITY(data-value)
specifies the name (1–16 characters) of the child activity to be resumed.

Conditions

107 ACTIVITYBUSY
RESP2 values:

19 The request timed out. It may be that another task using this activity-record has been prevented from ending.

109 ACTIVITYERR
RESP2 values:

8 The activity named on the ACTIVITY option could not be found.

14 The activity is in COMPLETE or CANCELLING mode, and therefore cannot be resumed.

16 INVREQ
RESP2 values:

4 The ACTIVITY option was used to name a child activity, but the command was issued outside the scope of a currently-active activity.

15 The ACQPROCESS option was used, but the unit of work that issued the request has not acquired a process.

24 The ACQACTIVITY option was used, but the unit of work that issued the request has not acquired an activity.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

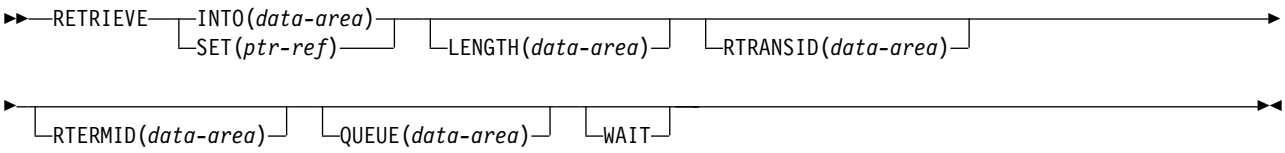
108 PROCESSERR
RESP2 values:

14 The process is in COMPLETE or CANCELLING mode, and therefore cannot be resumed.

RETRIEVE

Retrieve data stored for a task.

RETRIEVE



Conditions: ENDDATA, ENVDEFERR, INVREQ, IOERR, LENGERR

Note for dynamic transaction routing: Using RETRIEVE with WAIT could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. See Affinity for more information about transaction affinities.

Description

The RETRIEVE command retrieves data stored by expired START commands. It is the only method available for accessing such data.

A task that is not associated with a terminal can access only the single data record associated with the original START command; it does so by issuing a RETRIEVE command. The storage occupied by the data associated with the task is normally released on execution of the RETRIEVE command, or on termination of the task if no RETRIEVE command is executed before termination.

If the START command specified ATTACH, the storage is not released. (ASSIGN STARTCODE in such a task returns 'U' rather than 'S' or 'SD').

A task that is associated with a terminal can access all data records associated with all expired START commands having the same transaction identifier and terminal identifier as this task, that is the task issuing the RETRIEVE command; it does so by issuing consecutive RETRIEVE commands. Expired data records are presented to the task on request in expiration-time sequence, starting with any data stored by the command that started the task, and including data from any commands that have expired since the task started. Each data record is retrieved from temporary storage using the REQID of the original START command as the identification of the record in temporary storage.

When all expired data records have been retrieved, the ENDDATA condition occurs. The storage occupied by the single data record associated with a START

command is released after the data has been retrieved by a RETRIEVE command; any storage occupied by data that has not been retrieved is released when the CICS system is terminated.

If the retrieved data contains FMHs (Function Management Headers), as specified by the FMH option on the associated START command, field EIBFMH in the EIB is set to X'FF'. If no FMH is present, EIBFMH is set to X'00'.

Options

INTO(*data-area***)**
specifies the user data area into which retrieved data is to be written.

LENGTH(*data-area***)**
specifies a halfword binary value to define the length of the data area the retrieved data is written into.

If you specify the INTO option, the argument must be a data area that specifies the maximum length of data that the program is prepared to handle. If the value specified is less than zero, zero is assumed. If the length of the data exceeds the value specified, the data is truncated to that value and the LENGERR condition occurs. On completion of the retrieval operation, the data area is set to the original length of the data.

If you specify the SET option, the argument must be a data area. On completion of the retrieval operation, the data area is set to the length of the data.

For a description of a safe upper limit, see "LENGTH options in CICS commands" on page 10.

QUEUE(*data-area***)**
specifies the 8-character area for the temporary storage queue name that may be accessed by the transaction issuing the RETRIEVE command.

RTERMID(*data-area***)**
specifies a 4-character area that can be used in the TERMID option of a START command that may be executed subsequently.

RTRANSID(*data-area***)**
specifies a 4-character area that can be used in the TRANSID option of a START command that may be executed subsequently.

SET(*ptr-ref***)**
specifies the pointer reference to be set to the address of the retrieved data.

If DATALOCATION(ANY) is associated with the application program, the address of the data may be above or below the 16MB line.

If DATALOCATION(BELOW) is associated with the application program, and the data resides above the 16MB line, the data is copied below the 16MB line, and the address of this copy is returned.

If TASKDATAKEY(USER) is specified for the running task, and storage protection is active, the data returned is in a user-key. If TASKDATAKEY(CICS) is specified and storage protection is active, the data returned is in a CICS-key.

If you use SET you must also include LENGTH.

WAIT
specifies that, if all expired data records have already been retrieved, the task is to be put into a wait state until further expired data records become available. Although this means that the ENDDATA condition is not raised at

the time the RETRIEVE command is issued, it is raised later if CICS enters shutdown or if the task is subject to deadlock timeout and it waits for longer than the deadlock timeout interval. (See the DTIMOUT option of RDO DEFINE TRANSACTION.)

An attempt to issue RETRIEVE WAIT during shutdown leads to an AICB abend if there is no data record already available to satisfy the request.

If you use WAIT, you must have at least one other option.

Conditions

29 ENDDATA
occurs in any of the following situations:

- No more data is stored for the task issuing a RETRIEVE command. It can be considered a normal end-of-file response when retrieving data records sequentially.
- The RETRIEVE command is issued by a task that is started by a START command that did not specify any of the data options FROM, RTRANSID, RTERMID, or QUEUE.
- The RETRIEVE command is issued by a nonterminal task that was not created as a result of a START command.
- WAIT was specified and the task was waiting for a data record but none became available before the deadlock timeout interval (see the DTIMOUT option of RDO DEFINE TRANSACTION).
- WAIT was specified and the task was waiting when CICS entered shutdown. An attempt to issue RETRIEVE WAIT during shutdown leads to an AICB abend if there is no data record already available to satisfy the request.
- A RETRIEVE command with the WAIT option is issued when no data is available; the task was initiated by a START command that specified an APPC connection or terminal in the TERMID option.

Default action: terminate the task abnormally.

56 ENVDEFERR
occurs when a RETRIEVE command specifies an option not specified by the corresponding START command.

Default action: terminate the task abnormally.

16 INVREQ
occurs if the RETRIEVE command is not valid for processing by CICS.

Default action: terminate the task abnormally.

17 IOERR
occurs if an input/output error occurs during a RETRIEVE operation. The operation can be retried by reissuing the RETRIEVE command.

Default action: terminate the task abnormally.

22 LENGERR
occurs if the length specified is less than the actual length of the stored data.

Default action: terminate the task abnormally.

Examples

The following example shows how to retrieve data stored by a START command for the task, and store it in the user-supplied data area called DATAFLD.

```
EXEC CICS RETRIEVE
     INTO(DATAFLD)
     LENGTH(LENG)
```

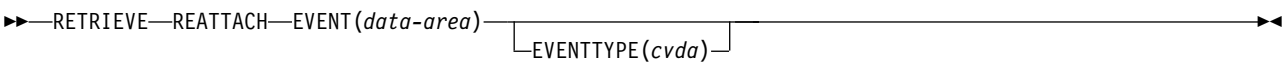
The following example shows how to request retrieval of a data record stored for a task into a data area provided by CICS; the pointer reference (PREF) specified by the SET option is set to the address of the storage area reserved for the data record.

```
EXEC CICS RETRIEVE
     SET(PREF)
     LENGTH(LENG)
```

RETRIEVE REATTACH EVENT

Retrieve the name of an event that caused the current BTS activity to be reattached.

RETRIEVE REATTACH EVENT



Conditions: END, INVREQ

Description

RETRIEVE REATTACH EVENT:

- Returns the name of the next event in the current BTS activity's reattachment queue.
- If the retrieved event is atomic, resets its fire status to NOTFIRED. (Composite events are not reset by this command, but only when their predicates become false.)

Use this command to find the name of the event that caused the activity to be reattached. In some cases, reattachment could result from the firing of more than one event—if, for example, the activity has previously been suspended, and reattachment events occurred while it was suspended; or if two or more timer events fire simultaneously. The event name or names are placed on the reattachment queue, from where they can be retrieved by issuing one or more RETRIEVE REATTACH EVENT commands.

Each time it is activated, an activity must deal with at least one reattachment event. That is, it must issue at least one RETRIEVE REATTACH EVENT command, and (if this is not done automatically by CICS) reset the fire status of the retrieved event to NOTFIRED—see Resetting and deleting reattachment events . Failure to

do so results in the activity completing abnormally, because it has made no progress—it has not reset any reattachment events and is therefore in danger of getting into an unintentional loop.

If there are multiple events on its reattachment queue, an activity can, by issuing multiple RETRIEVE REATTACH EVENT commands, deal with several or all of them in a single activation. Alternatively, it can deal with them singly, by issuing only one RETRIEVE command per activation and returning; it is then reactivated to deal with the next event on its reattachment queue. Which approach you choose is a matter of program design. Bear in mind, if you deal with several reattachment events in the same activation, that a syncpoint does not occur until the activation returns.

Note: The retrieval of a composite event from the reattachment queue does not reset the state of the composite event to NOTFIRED. Thus, if it retrieves a composite reattachment event, the activity program may need to issue one or more RETRIEVE SUBEVENT commands, to retrieve (and reset) the sub-event or sub-events that have fired. This in turn causes the fire status of the composite event to be re-evaluated.

Options

EVENT(data-area)
returns the 16-character name of the event which caused this activity to be reattached.

EVENTTYPE(cvda)
returns the type of the reattachment event. CVDA values are:

ACTIVITY
Activity completion.

COMPOSITE
Composite.

INPUT
Input

SYSTEM
The BTS system event, DFHINITIAL.

TIMER
Timer.

Conditions

83 END
RESP2 values:

8 There are no more events to retrieve.

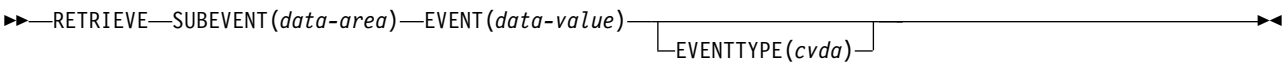
16 INVREQ
RESP2 values:

1 The command was issued outside the scope of an activity.

RETRIEVE SUBEVENT

Retrieve the name of the next sub-event in a BTS composite event's sub-event queue.

RETRIEVE SUBEVENT



Conditions: END, EVENTERR, INVREQ

Description

RETRIEVE SUBEVENT:

- Retrieves the name of the next sub-event in a BTS composite event's sub-event queue.
- Resets the retrieved sub-event's fire status to NOTFIRED.
- Causes the composite event's fire status to be re-evaluated.

The firing of a composite event results from the firing of a set of one or more sub-events. The names of sub-events that have fired are placed on the composite event's sub-event queue, from which they can be retrieved, in sequence, by issuing successive RETRIEVE SUBEVENT commands.

You can use this command to discover which sub-event or sub-events caused a composite event to fire.

Note:

1. The presence of events on the sub-event queue does not imply that the composite event has fired. (Some sub-events in the set required to fire the composite event may still be in NOTFIRED state, and not yet on the sub-event queue.) To discover whether a composite event has fired, use the TEST EVENT command.
2. Retrieval is destructive; when the name of a fired sub-event is retrieved, that sub-event cannot be retrieved again.
3. Because it resets the fire status of the sub-event, RETRIEVE SUBEVENT causes the fire status of the composite event to be re-evaluated.

Options

EVENT(data-value)
specifies the name (1–16 characters) of the composite event.

EVENTTYPE(cvda)
returns the type of the sub-event. CVDA values are:

ACTIVITY
Activity completion.

INPUT
Input

TIMER
Timer.

SUBEVENT(data-area)
returns the 16-character name of the sub-event at the head of the sub-event queue.

Conditions

83 END
RESP2 values:

9 There are no more sub-events to retrieve.

10 The composite event contains no sub-events (it is empty).

111 EVENTERR
RESP2 values:

4 The event specified on the EVENT option is not recognized by BTS.

16 INVREQ
RESP2 values:

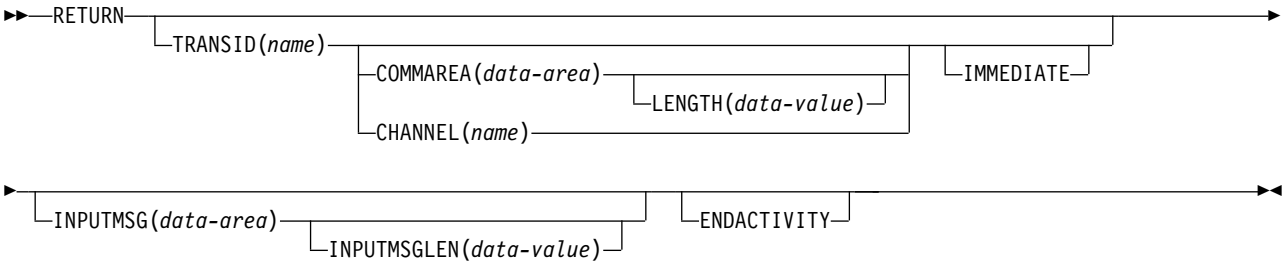
1 The command was issued outside the scope of an activity.

2 The event specified on the EVENT option is invalid. It is not a composite event.

RETURN

Return program control.

RETURN



Conditions: CHANNELERR, INVREQ, LENGERR

This command is threadsafe.

Description

RETURN returns control from an application program either to an application program at the next higher logical level, or to CICS.

When returning a communications area (COMMAREA), the LENGTH option specifies the length of the data to be passed. The LENGTH value being passed must not be greater than the length of the data area specified in the COMMAREA option. If it is, the results are unpredictable and may result in a LENGERR condition, as described in the section about passing data to other programs in Passing data to other programs.

The valid range for the COMMAREA length is 0 through 32 763 bytes. If the length provided is outside this range, the LENGERR condition occurs.

The COMMAREA, IMMEDIATE, and CHANNEL options can be used only when the RETURN command is returning control to CICS; otherwise, the INVREQ condition occurs.

No resource security checking occurs on the RETURN TRANSID command. However, transaction security checking is still available when CICS attaches the returned transaction.

Options

CHANNEL(*name***)**
specifies the name (1–16 characters) of a channel that is to be made available to the next program that receives control. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = ¬ , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters. If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

Channel names are always in EBCDIC. The allowable set of characters for channel names, listed above, includes some characters that do not have the same representation in all EBCDIC code pages. We therefore recommend that, if a channel is to be shipped between regions (that is, if the transaction named on the TRANSID option is remote), the characters used in naming it should be restricted to A-Z a-z 0-9 & : = , ; < > . - and _.

You can specify the channel name `DFHTRANSACTION` to use a transaction channel. A transaction channel does not go out of scope when the link level changes: it is always accessible in the transaction. For more information, see Channels and containers.

The program that issues the RETURN command can do one of the following:
- Have already created the channel by means of one or more PUT CONTAINER CHANNEL commands
- Specify its current channel, by name
- Name a non-existent channel, in which case a new, empty, channel is created

This option is valid only on a RETURN command issued by a program at the highest logical level; that is, a program returning control to CICS.

COMMAREA(*data-area***)**
specifies a communication area that is to be made available to the next program that receives control. In a COBOL receiving program, you must give this data area the name DFHCOMMAREA. See Sharing data across transactions for more information about the CICS COMMAREA. Because the data area is freed before the next program starts, a copy of the data area is created and a pointer to the copy is passed.

The communication area specified is passed to the next program that runs at the terminal. To ensure that the communication area is passed to the correct program, include the IMMEDIATE option.

This option is valid only on a RETURN command issued by a program at the highest logical level, that is, a program returning control to CICS.

ENDACTIVITY
This option is for use by programs that implement CICS business transaction services (BTS) activities. It specifies that the current activity is completing, and is not to be reactivated.

If there are no user events in the activity's event pool, the activity completes normally.

If there are user events (fired or unfired) in the activity's event pool:

- If one or more of the events are activity completion events, the activity abends. Trying to force an activity to complete before it has dealt with one or more of its child activities is a program logic error.
- If none of the events are activity completion events, the events are deleted and the activity completes normally.

For information about BTS in general and the ENDACTIVITY option in particular, see Activity completion.

This option is ignored outside the CICS BTS environment.

IMMEDIATE
ensures that the transaction specified in the TRANSID option is attached as the next transaction regardless of any other transactions enqueued by ATI for this terminal. The next transaction starts immediately and appears to the operator as having been started by terminal data. If the terminal is using bracket protocol, the terminal is also held in bracket. This option is valid only on a RETURN command issued by a program at the highest logical level, that is a program returning control to CICS.

Note that in a multi region environment, using IMMEDIATE does not affect the transaction definition as this is still found in the terminal-owning region (TOR).

INPUTMSG(*data-area*)
specifies data to be passed either to another transaction, identified by the TRANSID option, or to a calling program in a multiprogram transaction. You can also use INPUTMSG when returning control to CICS from a user-written dynamic transaction routing program, when you might want to modify the initial input.

In all cases, the data in the INPUTMSG data area is passed to the first program to issue a RECEIVE command following the RETURN.

See INPUTMSG for more information and illustrations about the use of INPUTMSG.

INPUTMSGLEN(*data-value*)
specifies a halfword binary value to be used with INPUTMSG.

LENGTH(*data-value*)
specifies a halfword binary value that is the length in bytes of the COMMAREA. For a description of a safe upper limit, see LENGTH options in CICS commands.

TRANSID(*name*)
specifies the transaction identifier (1–4 characters) to be used with the next input message entered from the terminal with which the task that issued the RETURN command has been associated. The specified name must have been defined as a transaction to CICS.

If TRANSID is specified for a program running on a terminal that is defined with a permanent transaction ID, the terminal's permanent transaction is initiated next rather than the transaction specified on the RETURN.

If you specify a TRANSID of binary zeros, the transaction identifier for the next program to be associated with the terminal may be determined from subsequent input from the terminal. Issuing a RETURN with a TRANSID of binary zeros and a COMMAREA can cause unpredictable results if the next transaction is not coded to handle the COMMAREA or if it receives a COMMAREA not intended for it.

If you specify TRANSID on a program that is not at the highest level, and there is a subsequent error on COMMAREA, INPUTMSG, or CHANNEL on the final RETURN, the TRANSID is cleared.

The next transaction identifier is also cleared on an abnormal termination of the transaction.

If IMMEDIATE is specified with this option, control is passed to the transaction specified in the TRANSID option in preference to any transactions enqueued by ATI.

If IMMEDIATE is not specified with this option, an ATI initiated transaction of the same name enqueued to the terminal nullifies this option.

This option is not valid if the transaction issuing the RETURN command is not associated with a terminal, or is associated with an APPC logical unit.

Conditions

122 CHANNELERR

RESP2 values:

1 The name specified on the CHANNEL option contains an illegal character or combination of characters.

16 INVREQ

RESP2 values:

1 A RETURN command with the TRANSID option is issued in a program that is not associated with a terminal.

2 A RETURN command with the CHANNEL, COMMAREA, or IMMEDIATE option is issued by a program that is not at the highest logical level.

4 A RETURN command with the TRANSID option is issued in a program that is associated with an APPC logical unit.

8 A RETURN command with the INPUTMSG option is issued for a program that is not associated with a terminal, or that is associated with an APPC logical unit, or an IRC session.

30 PG domain not initialized. Parameters are not allowed on the EXEC RETURN statement in first stage PLT programs.

200 A RETURN command is issued with an INPUTMSG option by a program invoked by DPL.

203 The CHANNEL option was specified but the remote region to which control is returned does not support channels.

Default action: terminate the task abnormally.

22 LENGERR
RESP2 values:

11 The COMMAREA length is less than 0 or greater than 32763.

26 The COMMAREA ADDRESS passed was zero, but the commarea length was non-zero.

27 The INPUTMSG LENGTH was less than 0 or greater than 32767.

Default action: terminate the task abnormally.

REWIND COUNTER and REWIND DCOUNTER

Rewind a named counter that has reached its limit (that is, the maximum number has been assigned). Use COUNTER for fullword signed counters and DCOUNTER for doubleword unsigned counters.

REWIND COUNTER



Conditions: BUSY, INVREQ, SUPPRESSED

This command is threadsafe.

REWIND DCOUNTER



Conditions: BUSY, INVREQ, SUPPRESSED

This command is threadsafe.

Description

These counter commands reset the current value of the named counter to its defined minimum number.

For information about specifying fullword and doubleword variables on these named counter commands, see "CICS command argument values" on page 3.

Options

COUNTER(*name***)**
Specifies the name of the named counter to reset to its minimum value. The name can be up to 16 alphanumeric characters. If *name* is a variable that contains a name that is less than 16 characters, the name must be padded with trailing blanks.

DCOUNTER(*name***)**
Specifies the name of the named counterto reset to its minimum value. The

name can be up to 16 alphanumeric characters. If *name* is a variable that contains a name that is less than 16 characters, the name must be padded with trailing blanks.

INCREMENT(*data-value***)**
Specifies, as a fullword signed binary value (or doubleword unsigned binary value for DCOUNTER), an increment to use to determine whether the named counter is in a valid state to be reset. If a previous GET command (that did not specify the REDUCE option) specified an increment that caused the GET command to fail, specify the same increment on the REWIND. The named counter server applies the increment before testing whether the counter is in a counter-at-limit condition.

See the INCREMENT option on the GET command for more details.

NOSUSPEND
Specifies an immediate return to the application program during a coupling facility structure rebuild. If you omit the NOSUSPEND parameter, the request is retried until it is successful. If you specify NOSUSPEND, the request fails immediately, and CICS returns BUSY, RESP2=500.

POOL(*poolname***)**
Specifies an 8-character string touse as a pool selection parameter to select the pool in which the named counter resides. The string can be a logical pool name, or the actual pool name.

Valid characters for the pool selector string are A through Z, 0 through 9, $ @ # and _ (underscore). If *name* is a variable that contains a name that is less than 8 characters, the name must be padded with trailing blanks.

This parameter is optional. If you omit the name of the pool, a pool selector value of 8 blanks is assumed.

If there is no matching entry in the DFHNCOPT options table, CICS uses the name specified on the NCPLDFT system initialization parameter, which specifies the default named counter pool.

For information about generating a named counter options table using the DFHNCO macro, see .

Conditions

128 BUSY
RESP2 values:

500 The NOSUSPEND option was specified on the command, and the coupling facility structure is not currently available during a rebuild.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

201 Named counter not found.

301 The server has reported an error code that is not understood by the named counter interface. Generally, this is not possible unless the interface load module, DFHNCIF, is at a lower maintenance or release level than the server itself.

303 An unexpected error, such as structure failure or loss of connectivity, has occurred on a macro used to access the coupling facility. Further information is in message DFHNC0441 in the application job log.

304 The pool selection parameter specified in the program cannot be resolved to a valid server name using the current options table.

305 The interface cannot establish a connection to the server for the selected named counter pool. Further information is in an AXM services message (AXMSC*nnnn*) in the application job log.

306 An abend occurred during server processing of a request. Further information is in a message in the application job log and the server job log.

308 The DFHNCOPT options table module, required to resolve a pool name, cannot be loaded.

309 During processing of the options table, the named counter interface encountered an unknown entry format. Either the options table is not correctly generated, or the DFHNCIF interface load module is not at the same release level as the options table.

310 An options table entry that matches the given pool name specified a user exit program, but the user exit program is not link-edited with the options table and cannot be loaded.

311 A response from the named counter server to the client region interface module, DFHNCIF, indicates that a system-managed rebuild is in progress but the EXEC CICS interface does not recognize the condition. This means that the CICS region is at CICS TS 2.1 or earlier.

403 The POOL parameter contains invalid characters or embedded spaces.

404 The COUNTER parameter contains invalid characters or embedded spaces.

406 The INCREMENT value is invalid. The value specified cannot be greater than the total range of the counter ((maximum value - minimum value) + 1).

Default action: terminate the task abnormally.

72 SUPPRESSED

RESP2 values:

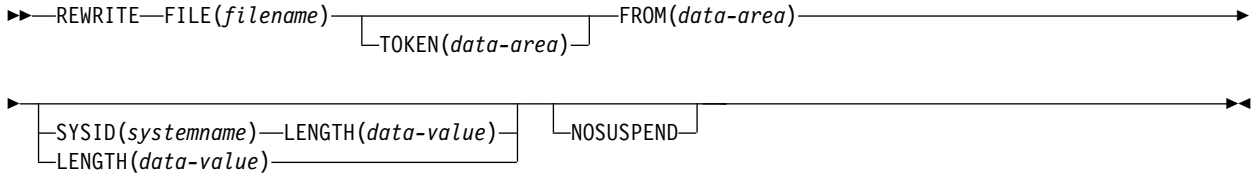
102 The named counter has not yet reached its limit (that is, the current value is not equal to the maximum value plus 1, giving the counter-at-limit condition). This error condition is returned if the named counter is not at its limit even after applying any specified increment.

Default action: terminate the task abnormally.

REWRITE

Update a record in a file.

REWRITE



Conditions: CHANGED, DUPREC, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, LENGERR, LOCKED, NOSPACE, NOTAUTH, NOTFND, RECORDBUSY, SYSIDERR

This command is threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over an IPIC connection to a remote CICS region.
- Defined as either local VSAM or RLS.

This command is not threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over a non-IPIC connection.
- Defined as a shared data table, coupling facility data table, or BDAM file.

Description

REWRITE updates a record in a file on a local or a remote system. You must always precede this command with a read with the UPDATE option.

For VSAM data sets, you must not change the key field in the record.

When this command is used to update a record in a CICS-maintained data table, the update is made to both the source VSAM KSDS and the in-memory data table. The details of the command for a CICS-maintained table are the same as for a VSAM KSDS.

When this command is used to update a record in a user-maintained data table, the update is made to the in-memory data table.

When this command is used to update records in a coupling facility data table, the update is made only to the data table in the coupling facility.

If an update to a user-maintained table or coupling facility data table does not succeed because there is insufficient space, a NOSPACE response is returned to the command. CICS also issues message DFHFC0432 to alert system administrators to the situation.

Options

FILE(*filename***)**
specifies the name of the file to be accessed.

If SYSID is specified, the data set to which this file refers is assumed to be on a remote system irrespective of whether the name is defined to CICS. Otherwise, the resource definition is used to find out whether the data set is on a local or a remote system.

FROM(*data-area***)**
specifies the record that is to be written to the data set referred to by this file.

LENGTH(*data-value***)**
specifies the length, as a halfword binary value, of the data area where the record is written from.

This option must be specified if SYSID is specified.

If the file is on a remote system, the LENGTH parameter need not be set here but must be set in the file resource definition.

If the file is on a local system, the LENGTH parameter must be set for variable-length records, using the INTO option, but not for fixed-length records. It is, however, advisable to specify the length for fixed-length records because it causes a check to be made that the record being written is not longer than that defined for the data set.

NOSUSPEND (RLS only)
The request does not wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK.

Lock contention can occur if the update involves changes made in RLS mode to records in a VSAM data set that has one or more alternate indexes, and an alternate index is defined with unique keys.

In most other cases, you should not need this option because the active lock is acquired when the task issues the **READ UPDATE** command.

Note: Requests that specify NOSUSPEND wait for at least 1 second before CICS returns the RECORDBUSY response.

SYSID(*systemname***)**
specifies the name of the system to which the request is directed.

TOKEN(*data-area***)**
specifies as a fullword binary value a unique request identifier for a REWRITE, used to associate it with a previous READ, READNEXT, or READPREV command that specified UPDATE.

TOKEN can be function shipped. However, if a request specifying TOKEN is function shipped to a member of the CICS family of products that does not support the TOKEN option, the request fails:

Conditions

105 CHANGED
RESP2 values:

109 A REWRITE command is issued for a file that is a defined as a coupling facility data table using the contention update model, and for which the record has been changed since the application program read it for update. To successfully update the record, repeat the READ for UPDATE to get the latest version of the record, reapply the change, and try the rewrite again.

Default action: terminate the task abnormally.

14 DUPREC
RESP2 values:

150 An attempt is made to rewrite a record to a data set whose upgrade

set has an alternate index with the UNIQUEKEY attribute, if the corresponding alternate key already exists in the alternate index.

Default action: terminate the task abnormally.

12 FILENOTFOUND

RESP2 values:

1 A file name referred to in the FILE option is not defined to CICS.

Default action: terminate the task abnormally.

21 ILLOGIC

Any browse that is currently in progress is terminated when this condition is raised.

RESP2 values: (VSAM)

110 A VSAM error occurs that is not in one of the other CICS response categories.

See EIBRCODE in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

16 INVREQ

RESP2 values:

30 A REWRITE command is issued without a token and no previous READ for UPDATE (also without a token) can be found.

A possible reason for the previous READ for UPDATE not being found is that it failed for some reason, and the failure has not been correctly handled or has been ignored.

46 A REWRITE command has attempted to change the length of a BDAM variable length record or block.

47 A REWRITE instruction includes a token whose value cannot be matched against any token that is in use for an existing READ for UPDATE request.

55 NOSUSPEND is not allowed because the file is not a VSAM file that is accessed in RLS mode.

56 An attempt to update a recoverable coupling facility data table has failed because the current unit of work has already updated 1024 recoverable coupling facility data tables. You cannot update more than 1024 recoverable coupling facility data tables within a unit of work

57 REWRITE operations are not allowed because the associated VSAM data set has an AVAILABILITY state of RREPL, which disallows such operations except those from replication programs.

Default action: terminate the task abnormally.

17 IOERR

RESP2 values:

120 An I/O error occurred during the file control operation. An I/O error is any unusual event that is not covered by a CICS condition.

For VSAM files, IOERR usually indicates a hardware error. Further information is available in the EXEC interface block; for details, see EIB fields.

For a coupling facility data table, an IOERR indicates a bad response returned from a coupling facility access.

Default action: terminate the task abnormally.

54 ISCINVREQ

RESP2 values:

70 The remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

22 LENGERR

RESP2 values:

10 The LENGTH option is not specified for a file with variable-length records, or for a BDAM file with undefined format records.

12 The length specified exceeds the maximum record size (of the source data set for a data-table); the record is truncated.

14 An incorrect length is specified for a file with fixed-length records.

Default action: terminate the task abnormally.

13 NOTFND

RESP2 values:

80 For user-maintained data tables, this condition occurs when an attempt to REWRITE a record has failed because the REWRITE is associated with a READ UPDATE request for a record that this transaction has deleted (using DELETE with RIDFLD) after it was read for update. This may be caused by a logic error in the application program.

This condition can also occur when a REWRITE command is issued to a coupling facility data table using the contention model and the record has been deleted since it was read for update.

Default action: terminate the task abnormally.

100 LOCKED

RESP2 values:

106 An attempt has been made to update a record, but a retained lock exists against a unique alternate key that is involved in the request.

Default action: abend the task with code AEX8.

18 NOSPACE

RESP2 values:

100 No space is available on the direct access device for adding the updated record to the data set.

102 The maximum number of records specified for a recoverable coupling facility data table has been exceeded. This can occur on a rewrite operation because an extra record is required in the coupling facility data table for recovery purposes until the update has been committed.

103 For user-maintained data tables, this condition occurs if CICS is unable to get sufficient storage in the CICS address space to store the updated data table entry.

108 For coupling facility data tables, this condition occurs if there is insufficient space in the coupling facility data table pool to store the updated record.

Default action: terminate the task abnormally.

70 NOTAUTH

RESP2 values:

101 A resource security check has failed on FILE(filename).

Default action: terminate the task abnormally.

101 RECORDBUSY

RESP2 values:

107 NOSUSPEND is specified but VSAM holds an active lock against a unique alternate index key that is involved in the request, which would cause the request to wait. See Retained and active locks for more information.

Default action: abend the task with code AEX9.

53 SYSIDERR

RESP2 values:

130 The SYSID option specifies a name that is neither the local system nor a remote system (made known to CICS by defining a CONNECTION). SYSIDERR also occurs when the link to the remote system is closed.

131 For a coupling facility data table, the connection to the coupling facility data table server has failed. This could be because the server itself has failed, or the server is available, but CICS has failed to connect to it.

132 The REWRITE is issued against a coupling facility data table that no longer exists, probably because of a coupling facility failure, in which case the coupling facility data table server also fails. See Setting up and running a coupling facility data table server for information about restarting a coupling facility data table server and reloading a table.

Default action: terminate the task abnormally.

Retained and active locks: RECORDBUSY refers to active locks and LOCKED refers to retained locks:

- REWRITE requests for records that have *retained* locks are always rejected with a LOCKED response.
- REWRITE requests for records that have *active* locks wait for the lock to be released, except when the NOSUSPEND keyword is specified, in which case CICS returns the RECORDBUSY response.

Examples

Here is an example of a simple REWRITE command:

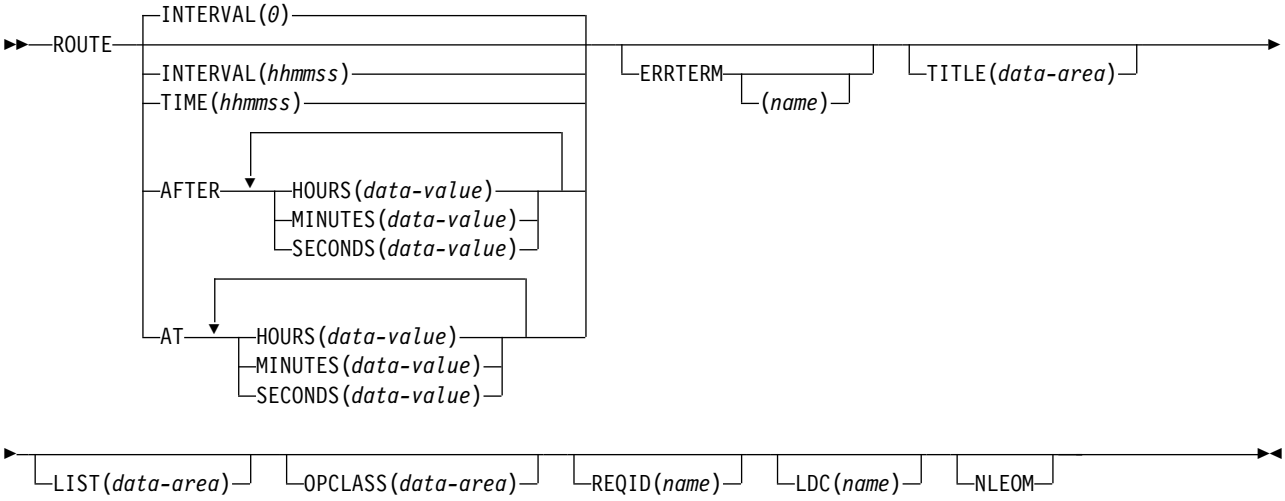
```
EXEC CICS REWRITE
     FROM(RECORD)
     FILE('MASTER')
     TOKEN(APTOK)
```

ROUTE

Route a BMS message. This command is only available on full BMS.

For more information about BMS, see Basic mapping support.

ROUTE



Conditions: IGREQID, INVERRTERM, INVLDC, INVREQ, RTEFAIL, RTESOME

Description

ROUTE routes a BMS logical message to one or more terminals or terminal operators.

The default is INTERVAL(0), but for C the default is AFTER HOURS(0) MINUTES(0) SECONDS(0).

Options

AFTER
specifies the amount of time to elapse before the route.

There are two ways to enter the time under AFTER and AT.

1. A combination of at least two of HOURS(0–99), MINUTES(0–59), and SECONDS(0–59). HOURS(1) SECONDS(3) would mean one hour and three seconds (the minutes default to zero).
2. As one of HOURS(0–99), MINUTES(0–5999), or SECONDS(0–359 999). HOURS(1) means one hour. MINUTES(62) means one hour and two minutes. SECONDS(3723) means one hour, two minutes, and three seconds.

AT
specifies the time of the route. For the ways to enter the time, see the AFTER option.

ERRTERM(*name*)
specifies the name of the terminal to be notified if the message is deleted

because it is undeliverable. The message number, title identification, and destination are indicated. If no name is specified, the originating terminal is assumed.

This option is effective only if PRGDLAY has been specified in the system initialization parameters.

HOURS(*data-value***)**
specifies a fullword binary value in the range 0–99. This is a suboption of the AFTER and AT options. For its use and meaning, see the AFTER option.

INTERVAL(*hhmmss***)**
specifies the interval of time after which the data is to be transmitted to the terminals specified in the ROUTE command. The **mm** and **ss** are in the range 0–59.

When using the C language, you are recommended to use the AFTER/AT HOURS, MINUTES, and SECONDS options as C does not provide a packed decimal data type. You may use INTERVAL, but if the value specified is **not** an integer constant, the application is responsible for ensuring that the value passed to CICS is in packed decimal format.

LDC(*name***) — logical units only**
specifies a 2-character mnemonic to be used to determine the logical device code (LDC) to be transmitted in the FMH to the logical unit. The mnemonic identifies an LDC entry defined by the DFHTCT TYPE=LDC macro.

When an LDC is specified, BMS uses the device type, the page size, and the page status associated with the LDC mnemonic to format the message. These values are taken from the extended local LDC table for the LU, if it has one. If the LU has only a local (unextended) LDC table, the values are taken from the system LDC table. The numeric value of the LDC is obtained from the local LDC table, unless this is an unextended table and the value is not specified, in which case it is taken from the system table.

If the LDC option is omitted, the LDC mnemonic specified in DFHMSD is used; see "DFHMSD" on page 877 for further details. If the LDC option has also been omitted from DFHMSD, the action depends on the type of logical unit, as follows:

3601 LU
The first entry in the local or extended local LDC table is used, if there is one. If a default cannot be obtained in this way, a null LDC numeric value (X'00') is used. The page size used is the value that is specified by the RDO TYPETERM options PAGESIZE or ALTPAGE, or (1,40) if such a value is not specified.

LUTYPE4 LU, batch LU, or batch data interchange LU
The local LDC table is not used to supply a default LDC; instead, the message is directed to the LU console. (Here, LU console means any medium on which the LU elects to receive such messages. For a batch data interchange LU, this does not imply sending an LDC in an FMH). The page size is obtained in the manner described for the 3601 LU.

For message routing, the LDC option of the ROUTE command takes precedence over all other sources. If this option is omitted and a route list is specified (LIST option), the LDC mnemonic in the route list is used; if the route list contains no LDC mnemonic, or no route list is specified, a default LDC is chosen as described above.

LIST(*data-area***)**
specifies the data area that contains a list of terminals and operators to which data is to be directed. If this option is omitted, all terminals supported by BMS receive the data (unless the OPCLASS option is in effect). See Route list format for the format of the route list.

MINUTES(*data-value***)**
specifies a fullword binary value in the range 0–59, when HOURS or SECONDS are also specified, or 0–5999 when MINUTES is the only option specified. This is a suboption of the AFTER and AT options. For its use and meaning, see the AFTER option.

NLEOM
specifies that data for a 3270 printer or a 3275 display with the printer adapter feature should be built with blanks and new-line (NL) characters, and that an end-of-message (EM) character should be placed at the end of the data. As the data is printed, each NL character causes printing to continue on the next line, and the EM character terminates printing.

The option is ignored if the device receiving the message (direct or routed) is not one of those mentioned above.

If this option is used, buffer updating and attribute modification of fields previously written into the buffer are not allowed. CICS includes the ERASE option with every write to the terminal.

The NL character occupies a buffer position. A number of buffer positions, equivalent to the value of the RDO options PAGESIZE or ALTPAGE for that terminal, are unavailable for data. This may cause data to wrap around in the buffer; if this occurs, the PAGESIZE or ALTPAGE value must be reduced.

OPCLASS(*data-area***)**
specifies the data area that contains a list of operator classes to which the data is to be routed. The classes are supplied in a 3-byte field, each bit position corresponding to one of the codes in the range 1 through 24 but in reverse order, that is, the first byte corresponds to codes 24 through 17, the second byte to codes 16 through 9, and the third byte to codes 8 through 1.

REQID(*name***)**
specifies a prefix (2–character field) to be used as part of a temporary storage identifier for CICS message recovery. Only one prefix can be specified for each logical message. The default prefix is **.

BMS message recovery is provided for a logical message only if the PAGING option is specified in the BMS SEND commands, and if the syncpoint has been reached.

SECONDS(*data-value***)**
specifies a fullword binary value in the range 0–59, when HOURS or MINUTES are also specified, or 0–359 999 when SECONDS is the only option specified. This is a suboption of the AFTER and AT options. For its use and meaning, see the AFTER option.

TIME(*hhmmss***)**
specifies the time of day at which data is to be transmitted to the terminals specified in the ROUTE command.

When using the C language, you are recommended to use the AFTER/AT HOURS, MINUTES, and SECONDS options as C does not provide a packed decimal data type. You may use TIME, but if the value specified is **not** an

integer constant, the application is responsible for ensuring that the value passed to CICS is in packed decimal format.

TITLE(*data-area***)**
specifies the data area that contains the title to be used with a routing logical message. This title appears as part of the response to a page query command. See Programming considerations with routing for the format of the title option.

Conditions

39 IGREQID
occurs if the prefix specified in the REQID option is different from that established by a previous REQID option, or by default for this logical message—REQID (**).

37 INVERRTERM
occurs if the terminal identifier specified in the ERRTERM option is not valid or is assigned to a type of terminal not supported by BMS.

Default action: terminate the task abnormally.

41 INVLDC
occurs if the specified LDC mnemonic is not included in the LDC list for the logical unit.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

4 Hours out of range

5 Minutes out of range

6 Seconds out of range

200 BMS commands are not supported for distributed program link.

also occurs (RESP2 not set) in the following situations:

- Bytes 10 through 15 of a route list entry do not contain blanks.

Default action: terminate the task abnormally.

33 RTEFAIL
occurs in any of the following situations:

- A ROUTE command would result in the message being sent only to. the terminal that initiated the transaction.
- A ROUTE command is issued against a remote. shippable terminal that is not yet installed in the application-owning region.

Default action: return control to the application program at the point immediately following the ROUTE command.

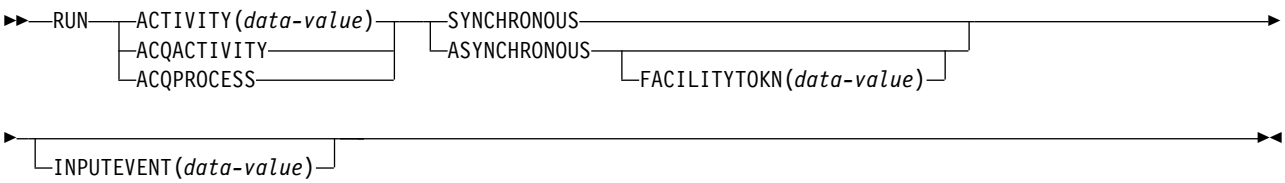
34 RTESOME
occurs if any of the terminals specified by the ROUTE command options do not receive the message.

Default action: return control to the application program at the point immediately following the ROUTE command.

RUN

Execute a CICS business transaction services process or activity synchronously or asynchronously, with context-switching.

RUN



Conditions: ACTIVITYBUSY, ACTIVITYERR, EVENTERR, INVREQ, IOERR, LOCKED, NOTAUTH, PROCESSBUSY, PROCESSERR

Description

RUN executes a CICS business transaction services process or activity synchronously or asynchronously with the requestor, with context-switching. The process or activity must previously have been defined to BTS.

RUN causes BTS to attach the process or activity, by sending it an input event. If the process or activity is in its initial state—that is, if this is the first time it is to be run, or if the activity has been reset by a RESET ACTIVITY command—CICS sends it the DFHINITIAL system event. If the process or activity is dormant—that is, waiting for a reattachment event to occur—the input event must be specified on the INPUTEVENT option.

If the process or activity is in any mode other than INITIAL or DORMANT, it cannot be run.

The SYNCHRONOUS and ASYNCHRONOUS options allow you to specify whether the process or activity should be executed synchronously or asynchronously with the requestor.

Context-switching

When a process or activity is activated by a RUN command, it is run:

- In a separate unit of work from the requestor.
- With the transaction attributes (TRANSID and USERID) specified on the DEFINE PROCESS or DEFINE ACTIVITY command.

In other words, a **context-switch** takes place. The relationship of the process or activity to the requestor is as between separate transactions, except that:

- Data can be passed between the two units of work
- The start and finish of the activity is related to the requestor's syncpoints.

To run a process or activity *without* context-switching—that is, in the same UOW and with the same TRANSID and USERID attributes as the requesting

transaction—use the LINK ACQPROCESS, LINK ACQACTIVITY, or LINK ACTIVITY command. This is possible only if the process or activity is run synchronously.

If the ability to isolate a failure is more important than performance, use RUN SYNCHRONOUS rather than LINK.

Activities

The only activities a program can run are as follows:

- If it is running as the activation of an activity, its own child activities. It can run several of its child activities within the same unit of work.
- The activity it has acquired, by means of an ACQUIRE ACTIVITYID command, in the current unit of work.

To check the response from the activity, the CHECK ACTIVITY command must be used. This is because the response to the request to run the activity does not contain any information about the success or failure of the activity itself—only about the success or failure of the request to run it.

Typically, if the activity is run synchronously, the CHECK command is issued immediately after the RUN command. If it is run asynchronously, the CHECK command could be issued:

- When the activity's parent is reattached due to the firing of the activity's completion event
- When the requestor is reattached due to the expiry of a timer.

The activity's completion event is one of the following:

1. The event named on the EVENT option of the DEFINE command for the activity.
2. If the DEFINE command did not specify a completion event, an event of the same name as the activity.

To retry an activity:

1. Issue a RESET ACTIVITY command to reset the activity to its initial state.
2. Issue a RUN command.

Processes

The only process that a program can run is the one that it has acquired in the current unit of work—see Acquiring processes and activities.

To check the response from the process, the CHECK ACQPROCESS command must be used. This is because the response to the request to run the process does not contain any information about the success or failure of the process itself—only about the success or failure of the request to run it.

Typically, if the process is run synchronously, the CHECK command is issued immediately after the RUN command. If the process is run asynchronously, the CHECK command could be issued when the requestor is reattached due to the expiry of a timer.

Options

ACQACTIVITY
specifies that the activity to be run is the one that the current unit of work has acquired by means of an ACQUIRE ACTIVITYID command.

ACQPROCESS
specifies that the process currently acquired by the requestor is to be run.

ACTIVITY(data-value)
specifies the name (1–16 characters) of the activity to be run. The name must be that of a child of the current activity.

ASYNCHRONOUS
specifies that the process or activity is to be executed asynchronously with the requestor.

FACILITYTOKN(data-value)
specifies an 8-byte bridge facility token.

This option applies when a BTS client activity runs a 3270-based pseudoconversational transaction. To ensure that the existing bridge facility is reused for the next transaction in the pseudoconversation, the client passes its token to the next child activity. This is explained in more detail in Reusing existing 3270 applications in BTS .

INPUTEVENT(data-value)
specifies the name (1–16 characters) of the event that causes the process or activity to be attached.

You *must not* specify this option if the process or activity is in its initial state; that is, if this is the first time it is to be run, or if the activity has been reset by a RESET ACTIVITY command. In this case, CICS sends the process or activity the DFHINITIAL system event.

You *must* specify this option if the process or activity is not in its initial state; that is, if it has been activated before, and has not been reset by a RESET ACTIVITY command.

If you specify INPUTEVENT, for the RUN command to be successful the process or activity to be attached must have defined the named event as an input event.

If you issue multiple asynchronous RUN commands against the same activity within the same unit of work:

- If you specify *the same input event*, each RUN command after the first fails.
- If you specify *different input events*, the activity may or may not be invoked as many times as the number of RUN requests—the only guarantee is that it will be invoked at least once. For example, if , within the same unit of work, you issue five asynchronous RUN requests for the same activity, specifying different input events, the activity might be invoked twice. At the first invocation, three input events might be presented, and at the second two.

SYNCHRONOUS
specifies that the process or activity is to be executed synchronously with the requestor.

Conditions

107 ACTIVITYBUSY
RESP2 values:

19 The request timed out. It may be that another task using this activity-record has been prevented from ending.

109 ACTIVITYERR
RESP2 values:

8 The activity named on the ACTIVITY option could not be found.

14 The activity to be run is not in INITIAL or DORMANT mode.

27 The activity named on the RUN SYNCHRONOUS command has abended.

111 EVENTERR
RESP2 values:

7 The event named on the INPUTEVENT option has not been defined by the activity or process to be run as an input event; or its fire status is FIRED.

16 INVREQ
RESP2 values:

4 The ACTIVITY option was used to name a child activity, but the command was issued outside the scope of a currently-active activity.

15 The task that issued the RUN ACQPROCESS command has not defined or acquired a process.

20 The SYNCHRONOUS option was used, but the activity to be run is suspended.

24 The ACQACTIVITY option was used, but the unit of work that issued the request has not acquired an activity.

28 CICS could not attach the transaction associated with the process or activity to be run. (This response occurs only on RUN SYNCHRONOUS commands.)

32 The SYNCHRONOUS option was used, but the transaction associated with the process or activity to be run is defined as remote. You cannot run a process or activity synchronously if its transaction is remote.

40 The program that implements the process or activity to be run is remote.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to run the process or activity.

106 PROCESSBUSY
RESP2 values:

13 The request timed out. It may be that another task using this process-record has been prevented from ending.

108 PROCESSERR
RESP2 values:

6 You cannot run the current process.

9 The process-type could not be found.

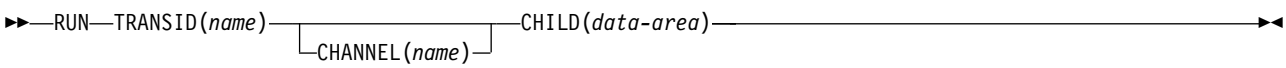
14 The process to be run is not in INITIAL or DORMANT mode.

27 The process named on the RUN SYNCHRONOUS command has abended.

RUN TRANSID

RUN TRANSID initiates a local child transaction that runs asynchronously with the parent transaction.

RUN TRANSID



Conditions: CHANNELERR, INVREQ, TRANSIDERR, NOTAUTH, DISABLED

Description

RUN TRANSID starts a task on the local system, optionally passing it a channel. The started task (child task) runs asynchronously with the starting task (parent task), and runs with the same security context as the parent task. The parent task uses the channel to pass input data to the child task. The parent task will get a response back when the API command **FETCH CHILD** or **FETCH ANY** is issued.

When a child task is started using **EXEC CICS RUN TRANSID**, it inherits the security context of the parent task. The security context is checked when **RUN TRANSID** is issued, and checked again at child task attach. If the child task does not have the authorization to run within the given context, it will abend with code AASA.

If the transaction (specified by TRANSID) is defined with DYNAMIC(YES), this option is ignored and no routing decision is made.

Options

CHANNEL(*name***)**
Specifies the name (1 - 16 characters) of a channel that is to be made available to the child task. The acceptable characters are A-Z a-z 0-9 $ @ # ./-_ % & ? ! : | " = ¬ , ; < >. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters. If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

Channel names are always in EBCDIC. The set of allowed characters for channel names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be

shipped between regions, it is advisable to restrict the characters used to name them to A-Z a-z 0-9 & : = , ; < > . - and _.

You can specify the channel name DFHTRANSACTION to use a transaction channel. A transaction channel does not go out of scope when the link level changes: it is always accessible in the transaction. For more information, see Channels and containers.

The program that issues the **RUN TRANSID** command can do one of the following:

- Create the channel before issuing the **RUN TRANSID** command, by using one or more **PUT CONTAINER CHANNEL** or **PUT64 CONTAINER** commands.
- Specify its current channel, by name.
- Name a channel that does not currently exist. A new empty channel is created.

The child task is given a *copy* of the channel's containers (and the data that they contain). The copy is made when the **RUN TRANSID** command is issued.

CHILD(*data-area***)**
Specifies a 16-character binary data area, into which CICS will place the child token that represents the child task. The child token is used to refer to the child task in order to get a response back from it through an API command such as FETCH CHILD.

TRANSID(*name***)**
Specifies the symbolic identifier (1–4 characters) of the local transaction to be started as the result of a RUN command.

Conditions

122 CHANNELERR
RESP2 values:

1 The channel specified on the CHANNEL option contains an incorrect character or combination of characters.

16 INVREQ
RESP2 values:

17 The started transaction is not shutdown-enabled, and the CICS region is in the process of shutting down. INVREQ also occurs (RESP2 not set) if the RUN command is not valid for processing by CICS.

19 RUN TRANSID was issued during transaction termination.

28 TRANSIDERR
RESP2 values:

1 The transaction identifier specified in a RUN TRANSID command is not defined to CICS.

11 The specified transaction is defined as remote.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to run the child task.

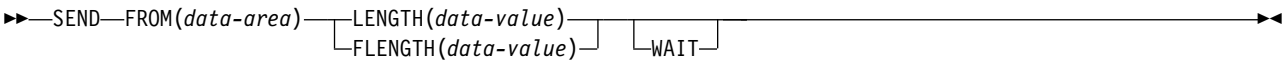
84 DISABLED
RESP2 values:

50 The specified transaction is disabled.

SEND (z/OS Communications Server default)

Write data to a standard CICS supported terminal.

SEND (z/OS Communications Server default)



Conditions: INVREQ, LENGERR, NOTALLOC

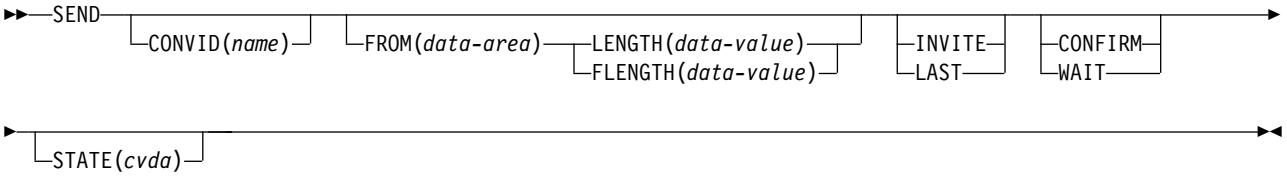
Description

SEND writes data to a terminal. This form of the send command can be used by all CICS-supported terminals for which the other SEND descriptions are not appropriate.

SEND (APPC)

Send data on an APPC mapped conversation.

SEND (APPC)



Conditions: INVREQ, LENGERR, NOTALLOC, SIGNAL, TERMERR

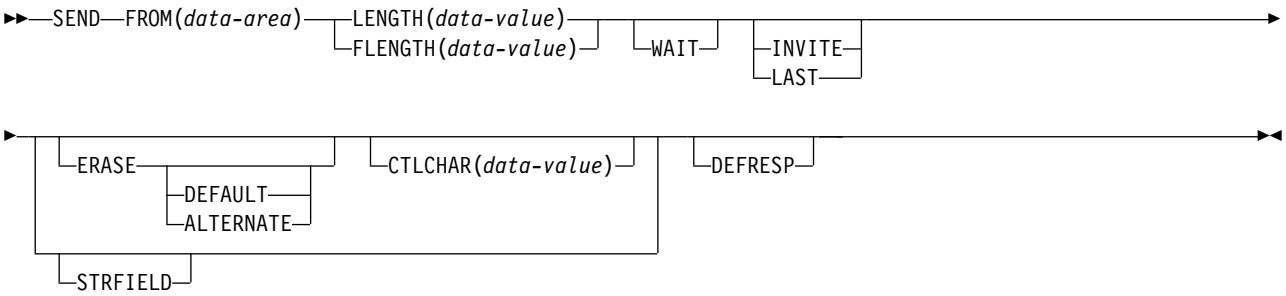
Description

SEND sends data to a conversation partner on an APPC mapped conversation.

SEND (LUTYPE2/LUTYPE3)

Write data to a 3270-display logical unit (LUTYPE2) or a 3270-printer logical unit (LUTYPE3).

SEND (LUTYPE2/LUTYPE3)



Conditions: INVREQ, LENGERR, TERMERR

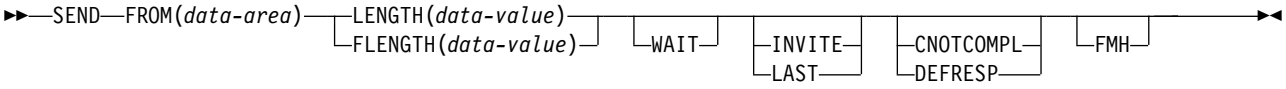
Description

The **SEND** command writes data to a terminal.

SEND (LUTYPE4)

Write data to a LUTYPE4 logical unit.

SEND (LUTYPE4)



Conditions: INVREQ, IGREQCD, LENGERR, SIGNAL, TERMERR

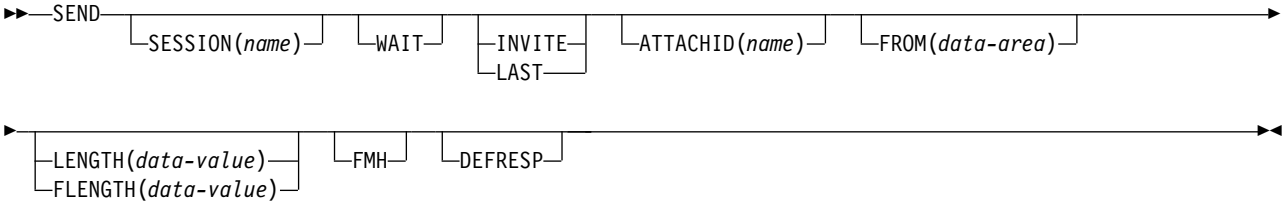
Description

SEND writes data to a terminal.

SEND (LUTYPE6.1)

Send data on an LUTYPE6.1 conversation.

SEND (LUTYPE6.1)



Conditions: CBIDERR, INVREQ, LENGERR, NOTALLOC, SIGNAL, TERMERR

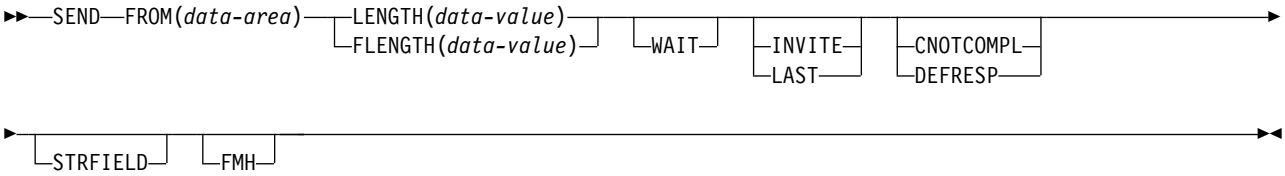
Description

SEND sends data to a conversation partner on an LUTYPE6.1 conversation.

SEND (SCS)

Write data to a 3270 SCS printer logical unit.

SEND (SCS)



Conditions: INVREQ, LENGERR, TERMERR

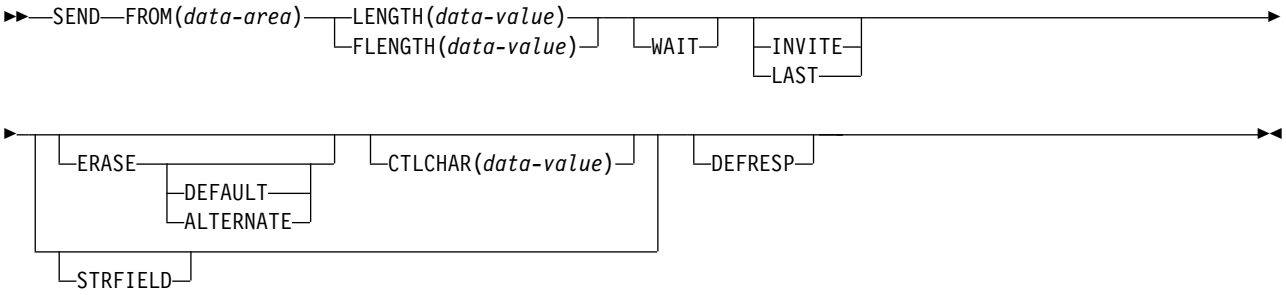
Description

SEND writes data to a logical unit. The SCS printer logical unit accepts a character string as defined by Systems Network Architecture (SNA).

SEND (3270 logical)

Write data to a 3270 logical unit.

SEND (3270 logical)



Conditions: INVREQ, LENGERR, TERMERR

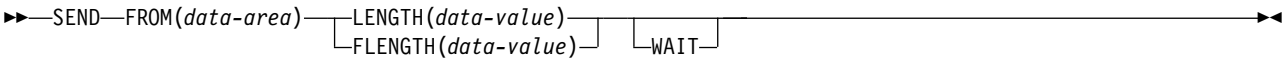
Description

SEND writes data to a terminal.

SEND (3600 pipeline)

Write data to a 3600 pipeline logical unit.

SEND (3600 pipeline)



Conditions: INVREQ, LENGERR, TERMERR

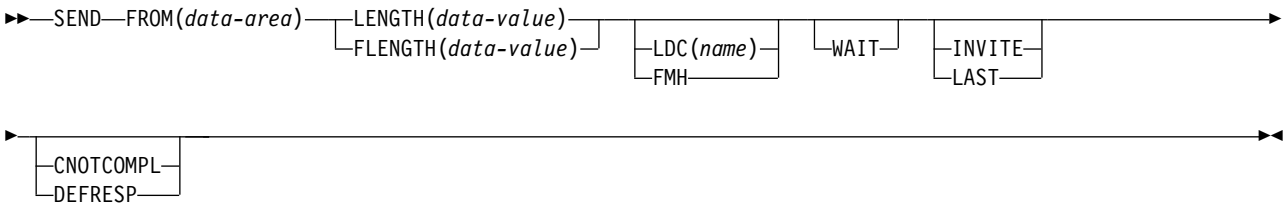
Description

SEND writes data to a terminal.

SEND (3600-3601)

Write data to a 3600 (3601) logical unit.

SEND (3600-3601)



Conditions: INVREQ, LENGERR, SIGNAL, TERMERR

Description

SEND writes data to a terminal. This form of SEND also applies to the 4770 and the 3630 plant communication system.

A logical device code (LDC) is a code that can be included in an outbound FMH to specify the disposition of the data (for example, to which subsystem terminal it should be sent). Each code can be represented by a unique LDC mnemonic.

The installation can specify up to 256 2-character mnemonics for each TCTTE, and two or more TCTTEs can share a list of these mnemonics. Corresponding to each LDC mnemonic for each TCTTE is a numeric value (0 through 255).

A 3600 device and a logical page size are also associated with an LDC. LDC or *LDC value* is used in this information to refer to the code specified by the user. *LDC mnemonic* refers to the 2-character symbol that represents the LDC numeric value.

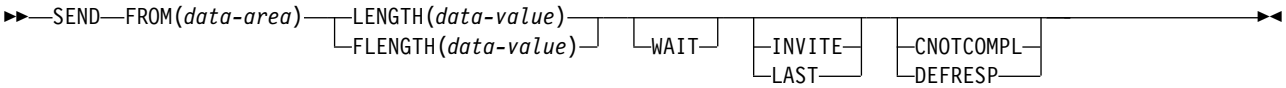
When the LDC option is specified, the numeric value associated with the mnemonic for the particular TCTTE, is inserted in the FMH. The numeric value associated with the LDC mnemonic is chosen by the installation, and is interpreted by the 3601 application program.

On output, the FMH can be built by the application program or by CICS. If your program supplies the FMH, you place it at the front of your output data and specify the FMH option on your SEND command. If you omit the FMH option, CICS will provide an FMH but you must reserve the first three bytes of the message for CICS to complete.

SEND (3600-3614)

Write data to a 3600 (3614) logical unit.

SEND (3600-3614)



Conditions: INVREQ, LENGERR, TERMERR

Description

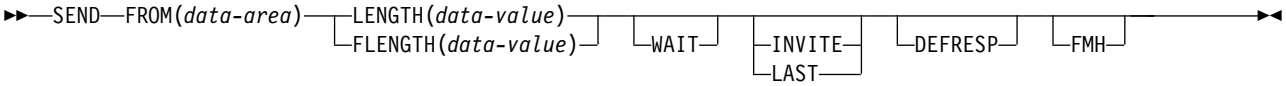
SEND writes data to a terminal. The data stream and communication format used between a CICS application program and a 3614 is determined by the 3614. The application program is therefore device dependent when handling 3614 communication.

For further information about designing 3614 application programs for CICS, refer to the *IBM 4700/3600/3630 Guide*.

SEND (3650 interpreter)

Write data to a 3650 interpreter logical unit.

SEND (3650 interpreter)



Conditions: INVREQ, LENGERR, TERMERR

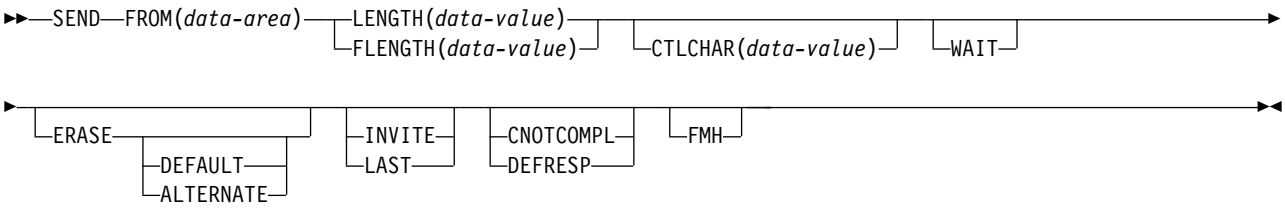
Description

SEND writes data to a terminal.

SEND (3650-3270)

Write data to a 3650 logical unit.

SEND (3650-3270)



Conditions: INVREQ, LENGERR, TERMERR

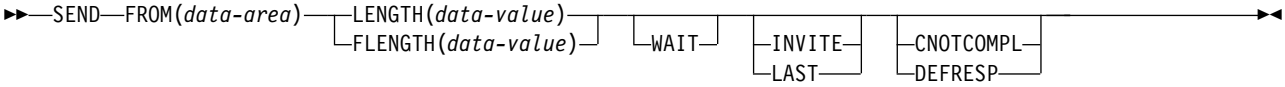
Description

SEND writes data to a terminal.

SEND (3650-3653)

Write data to a 3650 (3653) logical unit.

SEND (3650-3653)



Conditions: INVREQ, LENGERR, TERMERR

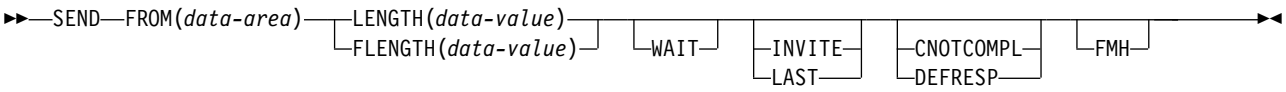
Description

SEND writes data to a terminal.

SEND (3650-3680)

Write data to a 3650 (3680) logical unit.

SEND (3650-3680)



Conditions: INVREQ, LENGERR, TERMERR

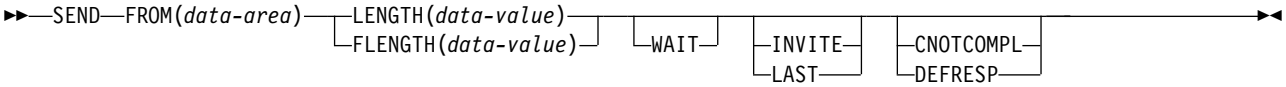
Description

SEND writes data to a terminal.

SEND (3767)

Write data to a 3767 interactive logical unit.

SEND (3767)



Conditions: INVREQ, LENGERR, SIGNAL, TERMERR

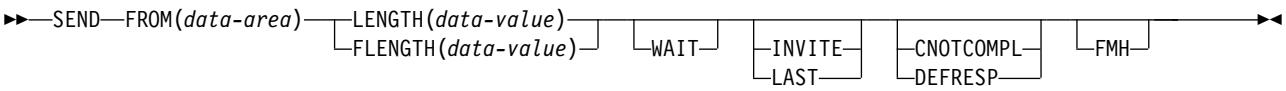
Description

SEND writes data to a terminal. This form of SEND also applies to the 3770 interactive logical unit.

SEND (3770)

Write data to a 3770 batch logical unit.

SEND (3770)



Conditions: INVREQ, LENGERR, SIGNAL, TERMERR

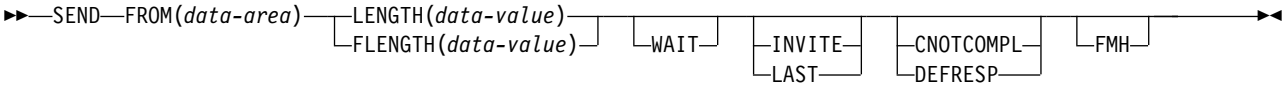
Description

SEND writes data to a terminal.

SEND (3790 full-function or inquiry)

Write data to a 3790 full-function or inquiry logical unit.

SEND (3790 full-function or inquiry)



Conditions: INVREQ, LENGERR, SIGNAL, TERMERR

Description

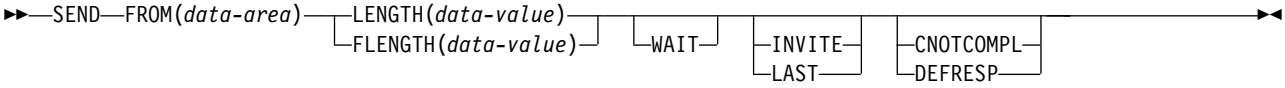
SEND writes data to a terminal. This form of SEND also applies to the following devices:

- 3650/3680 full-function logical unit
- 3770 full-function logical unit

SEND (3790 SCS)

Write data to a 3790 SCS printer logical unit.

SEND (3790 SCS)



Conditions: INVREQ, LENGERR, TERMERR

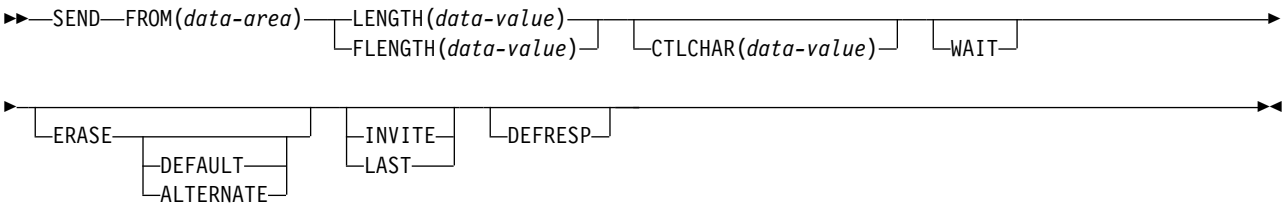
Description

SEND writes data to a terminal.

SEND (3790 3270-display)

Write data to a 3790 (3270-display) logical unit.

SEND (3790 3270-display)



Conditions: INVREQ, LENGERR, TERMERR

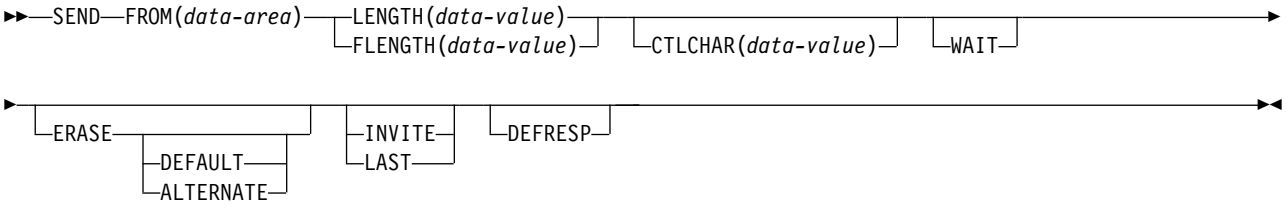
Description

SEND writes data to a terminal.

SEND (3790 3270-printer)

Write data to a 3790 (3270-printer) logical unit.

SEND (3790 3270-printer)



Conditions: INVREQ, LENGERR, TERMERR

Description

SEND writes data to a terminal.

SEND: z/OS Communications Server options

Common options used on the SEND command (z/OS Communications Server).

Options

ALTERNATE
Sets the terminal to use the ALTERNATE screen size.

ATTACHID(*name***)**
Specifies that an attach header (created by a **BUILD ATTACH** command) is to precede, and be concatenated with, the user data supplied in the FROM option. "name" (1–8 characters) identifies the attach header control block to be used in the local task.

CNOTCOMPL
Indicates that the request/response unit (RU) sent as a result of this **SEND** command does not complete the chain. If this option is omitted and chain assembly is specified, the RU terminates the chain.

CONFIRM
Indicates that an application using a synchronization level 1 or 2 conversation requires a response from the remote application. A remote CICS application can respond positively by running an **ISSUE CONFIRMATION** command, or negatively, by running an **ISSUE ERROR** command, in which case the sending application has EIBERR and EIBERRCD set. CICS does not return control to the sending application until the response is received.

CONVID(*name***)**
Identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previous **ALLOCATE** command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previous **ASSIGN** command).

For compatibility with earlier releases, SESSION is accepted as a synonym for CONVID. New programs must use CONVID.

If this option is omitted, the principal facility is assumed.

CTLCHAR(*data-value***)**
Specifies a 1-byte write control character (WCC) that controls a **SEND** command for a 3270. These characters are described in IBM 3270 Data Stream Programmers Reference. A COBOL user must specify a data area containing this character. If the option is omitted, all modified data tags are reset to zero and the keyboard is restored.

DEFAULT
Sets the terminal to use the DEFAULT screen size.

DEFRESP
Indicates that a definite response is required when the output operation completes.

ERASE
Specifies that the screen printer buffer or partition is to be erased and the cursor returned to the upper left corner of the screen. This option applies only to the 3270, or 8775, and to the 3604 Keyboard Display.

The first output operation in any transaction, or in a series of pseudoconversational transactions, must always specify the ERASE option. For transactions attached to 3270 screens or printers, unless explicitly overridden by the DEFAULT or ALTERNATE option, this options also ensures that the correct screen size is selected, as defined for the transaction by the SCRNSIZE option in the PROFILE resource definition.

FLENGTH(*data-value***)**
An alternative to the LENGTH option. For architectural reasons, this option is limited to a maximum of 32 KB for all terminal-related **SEND** commands.

FMH
Specifies that a function management header is included in the data to be written. If the ATTACHID option is specified as well, the concatenated FMH flag is set in the attach FMH.

The use of FMH is optional and is not supported for all terminal types. If not supplied, CICS takes no action, except for 3600/4700 terminals, where an FMH is mandatory. In this case, if FMH is not specified, CICS supplies one and places it in the first 3 bytes of the message, which you must reserve for this purpose.

FROM(*data-area***)**
Specifies the data to be written to the logical unit, or a partner transaction.

INVITE
For the **SEND** (APPC) command, an application can use the INVITE option to add a change-direction indicator to data already sent to a process in a connected APPC system. Control data is not transmitted by CICS until the subsequent WAIT or a **SYNCPOINT** command, unless CONFIRM or WAIT is also coded on the **GDS SEND INVITE** command.

For the other **SEND** commands, INVITE specifies that the next terminal control command to be run for this facility is a RECEIVE. This allows optimal flows to occur.

LAST
Specifies that this is the last **SEND** command for a transaction.

LDC(*name***)**
Specifies the 2-character mnemonic used to determine the appropriate logical device code (LDC) numeric value. The mnemonic represents an LDC entry in the terminal control table TYPE=LDC.

LENGTH(*data-value***)**
Specifies the length, as a halfword binary value, of the data to be written. For a description of a safe upper limit, see "LENGTH options in CICS commands" on page 10.

SESSION(*name***)**
Specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used. If this option is omitted, the principal facility for the task is used.

STATE(*cvda***)**
Gets the state of the current conversation. The CVDA values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE

- SYNCRECEIVE
- SYNCSEND

STRFIELD
Specifies that the data area specified in the FROM option contains structured fields. If this option is specified, the contents of all structured fields must be handled by the application program. The **CONVERSE** command, rather than a **SEND** command, must be used if the data area contains a read partition structured field. Structured fields are described in IBM 3270 Data Stream Device Guide.)

The CTLCHAR and ERASE options are mutually exclusive with STRFIELD, and their use with STRFIELD generates an error message.

WAIT
Specifies that processing of the command must be completed before any subsequent processing is attempted.

If the WAIT option is not specified, control is returned to the application program when processing of the command starts. A subsequent input or output request (terminal control, BMS, or batch data interchange) to the terminal associated with the task causes the application program to wait until the previous request completes.

Conditions

Some of the following conditions might occur in combination. If more than one occurs, only the first is passed to the application program. EIBRCODE, however, is set to indicate all the conditions that occurred.

62 CBIDERR
Occurs if the requested attach header control block named in ATTACHID cannot be found.

Default action: terminate the task abnormally.

57 IGREQCD
Occurs when an attempt is made to run a **SEND** command after a SIGNAL data flow control command with a request change direction (RCD) code has been received from the logical unit.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application attempted to send on its function-shipping session (its principal facility).

For **SEND** (APPC), a distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Also occurs (RESP2 not set) in any of the following situations:

- The CONFIRM option has been specified, but the APPC conversation is not sync level 1 or 2.
- The SEND command has been used on an APPC conversation that is not a mapped conversation or that is not using the EXEC CICS interface.

Default action: terminate the task abnormally.

22 LENGERR
Occurs if an out-of-range value is supplied in the LENGTH or FLENGTH option.

Default action: terminate the task abnormally.

61 NOTALLOC
Occurs if the CONVID value in the command does not relate to a conversation that is owned by the application, or if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

24 SIGNAL
Occurs when an inbound SIGNAL data flow control command is received from a logical unit or session. EIBSIG is always set when an inbound signal is received.

Default action: ignore the condition.

81 TERMERR
Occurs for a session-related error. Any action on that conversation other than a FREE causes an ATCV abend.

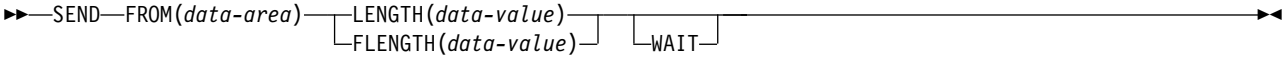
A CANCEL TASK request by a user node error program (NEP) might cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

SEND (non-z/OS Communications Server default)

Write data to standard CICS terminal support.

SEND (non-z/OS Communications Server default)



Conditions: INVREQ, LENGERR, NOTALLOC

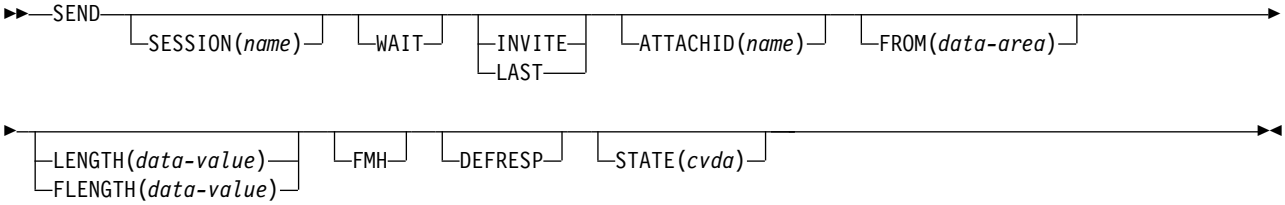
Description

SEND writes data to a terminal. This form of the send command can be used by all CICS-supported terminals for which the other SEND descriptions are not appropriate.

SEND (MRO)

Send data on an MRO conversation.

SEND (MRO)



Conditions: CBIDERR, INVREQ, LENGERR, NOTALLOC, TERMERR

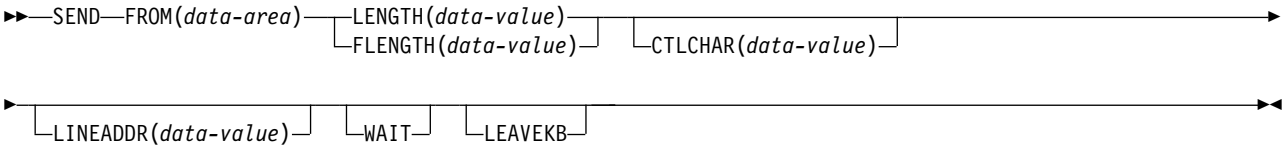
Description

SEND sends data to a conversation partner on an MRO conversation.

SEND (2260)

Write data to a 2260 or 2265 display station.

SEND (2260)



Conditions: INVREQ, LENGERR

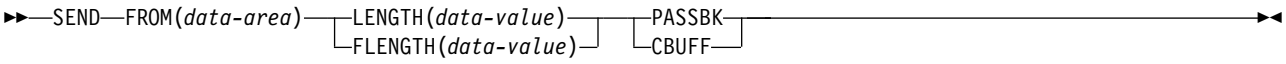
Description

SEND writes data to a terminal.

SEND (2980)

Write data to a 2980 general banking terminal system.

SEND (2980)



Conditions: INVREQ, LENGERR, NOPASSBKWR

Description

SEND writes data to a terminal. For more information about the 2980 general banking system, see "RECEIVE (2980)" on page 424.

SEND: non-z/OS Communications Server options

Common options used on the SEND command (non-z/OS Communications Server).

Options

ALTERNATE
sets the terminal to use the ALTERNATE screen size.

ASIS
indicates that output is to be sent in transparent mode (with no recognition of control characters and accepting any of the 256 possible combinations of eight bits as valid transmittable data).

Note: If you are using a katakana terminal, you might see some messages containing mixed English and katakana characters. That is because katakana terminals cannot display mixed-case output. Uppercase characters in the data stream appear as uppercase English characters, but lowercase characters appear as katakana characters. If this happens, ask your system programmer to specify MSGCASE=UPPER in the system initialization parameters so that messages contain uppercase characters only.

ATTACHID(*name***)**
specifies that an attach header (created by a BUILD ATTACH command) is to precede, and be concatenated with, the user data supplied in the FROM option. "name" (1–8 characters) identifies the attach header control block to be used in the local task.

CBUFF
specifies that data is to be written to a common buffer in a 2972 control unit. The WAIT option is implied.

CNOTCOMPL
indicates that the request/response unit (RU) sent as a result of this SEND command does not complete the chain. If this option is omitted and chain assembly has been specified, the RU terminates the chain.

CTLCHAR(*data-value***)**
specifies a 1-byte write control character (WCC) that controls a SEND command for a 3270. These are documented in the *IBM 3270 Data Stream Programmer's Reference*. A COBOL user must specify a data area containing this character. If the option is omitted, all modified data tags are reset to zero and the keyboard is restored.

DEFAULT
sets the terminal to use the DEFAULT screen size.

DEFRESP
indicates that a definite response is required when the output operation has been completed.

ERASE
specifies that the screen printer buffer or partition is to be erased and the cursor returned to the upper left corner of the screen. (This option applies only to the 3270, or 8775, and to the 3604 Keyboard Display.)

The first output operation in any transaction, or in a series of pseudoconversational transactions, should always specify ERASE. For transactions attached to 3270 screens or printers, unless explicitly overridden by the DEFAULT or ALTERNATE option, this also ensures that the correct screen size is selected, as defined for the transaction by the SCRNSIZE option in the RDO PROFILE resource definition.

FLENGTH(*data-value***)**
A fullword alternative to LENGTH.

FMH
specifies that a function management header has been included in the data in the FROM area. If the ATTACHID option is specified as well, the concatenated FMH flag is set in the attach FMH.

FROM(*data-area***)**
specifies the data to be written to the logical unit or terminal.

INVITE
specifies that the next terminal control command to be executed for this facility is a RECEIVE. This allows optimal flows to occur.

LAST
specifies that this is the last output operation for a transaction and therefore the end of a bracket.

LEAVEKB
specifies that the keyboard is to remain locked at the completion of the data transfer.

LENGTH(*data-value***)**
specifies the length, as a halfword binary value, of the data to be written. For a description of a safe upper limit, see “LENGTH options in CICS commands” on page 10.

LINEADDR(*data-value***)**
specifies that the writing is to begin on a specific line of a 2260/2265 screen. The data value is a halfword binary value in the range 1 through 12 for a 2260, or 1 through 15 for a 2265.

PASSBK
specifies that communication is with a passbook. The WAIT option is implied.

PSEUDOBIN (*start-stop only***)**
specifies that the data being written is to be translated from System/7 hexadecimal to pseudobinary.

SESSION(*name***)**
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used. If this option is omitted, the principal facility for the task is used.

STATE(*cvda*)
gets the state of the transaction program. The cvda values returned by CICS are:

- ALLOCATED
- FREE
- PENDFREE
- RECEIVE
- ROLLBACK
- SEND
- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

STRFIELD
specifies that the data area specified in the FROM option contains structured fields. If this option is specified, the contents of all structured fields must be handled by the application program. The CONVERSE command, rather than a SEND command, must be used if the data area contains a read partition structured field. (Structured fields are described in the *CICS 3270 Data Stream Device Guide*.)

CTLCHAR and ERASE are mutually exclusive with STRFIELD, and their use with STRFIELD generates an error message.

WAIT
specifies that processing of the command must be completed before any subsequent processing is attempted.

If the WAIT option is not specified, control is returned to the application program when processing of the command has started. A subsequent input or output request (terminal control, BMS, or batch data interchange) to the terminal associated with the task causes the application program to wait until the previous request has been completed.

Conditions

62 CBIDERR
occurs if the requested attach header control block named in ATTACHID cannot be found.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 occurs if a distributed program link server application attempted to send on its function-shipping session (its principal facility)

Default action: terminate the task abnormally.

22 LENGERR
occurs if an out-of-range value is supplied in the LENGTH or FLENGTH option.

Default action: terminate the task abnormally.

51 NOPASSBKWR
occurs if no passbook is present.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

81 TERMERR
occurs for a terminal-related error, such as a session failure. This condition applies to z/OS Communications Server-connected terminals only.

A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program (DFHZNAC) handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

03 WRBRK
occurs if the command is terminated by the attention key.

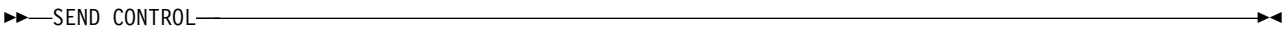
Default action: ignore the condition.

SEND CONTROL

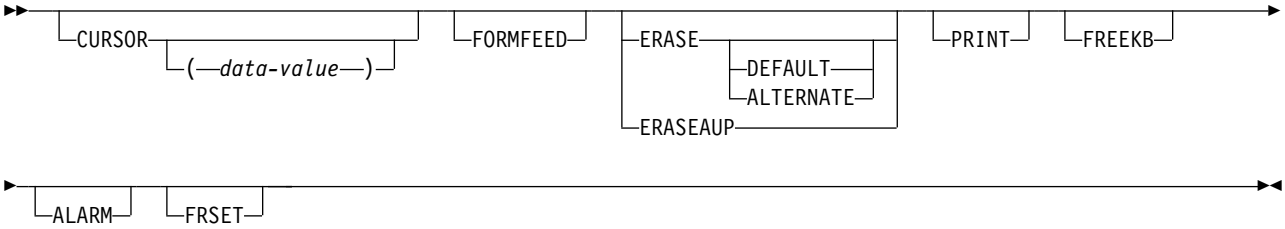
Send device controls to a terminal without map or text data. The keywords are separated into those supported by minimum, standard, and full BMS.

For further information about BMS, see Basic mapping support.

SEND CONTROL



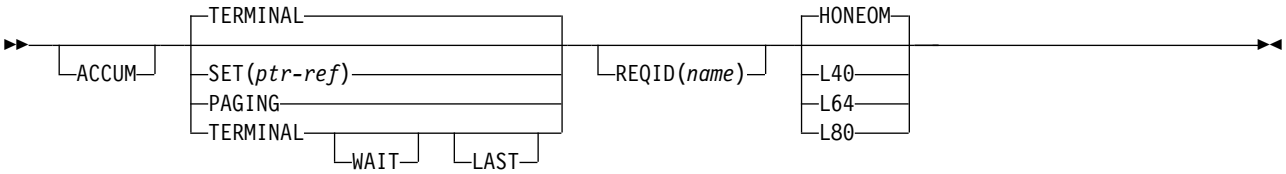
SEND CONTROL Minimum BMS



SEND CONTROL Standard BMS



SEND CONTROL Full BMS



Conditions: IGREQCD, IGREQID, INVLDC, INVPARTN, INVREQ, RETPAGE, TSIOERR, WRBRK

Description

SEND CONTROL sends device controls to a terminal.

When using the SEND CONTROL command with any of the ALARM, FREEKB, FRSET, HONEOM, L40, L64, L80, or PRINT options, see DFHMDI options, CTRL for a description of the option priority.

ACCUM
specifies that this command is one of a number of commands that are used to build a logical message. The logical message is completed by a SEND PAGE command, or deleted by a PURGE MESSAGE command.

ACTPARTN(*name*)
specifies the name (1–2 characters) of the partition to be activated. Activating a partition moves the cursor into the specified partition, and unlocks the keyboard for the specified partition.

This option is ignored if the target terminal does not support partitions, or if there is no application partition set.

ALARM
specifies that the 3270 audible alarm feature is to be activated. For logical units supporting FMHs (except interactive and batch logical units), ALARM instructs BMS to set the alarm flag in the FMH.

ALTERNATE
sets the terminal to use the ALTERNATE screen size.

CURSOR(*data-value*)
specifies the location the 3270 or 3604 cursor is returned to on completion of a SEND CONTROL command.

The data value must be a halfword binary value that specifies the cursor position relative to zero; the range of values that can be specified depends on the size of the screen being used.

If ACCUM is being used, the most recent value of CURSOR specified is used to position the cursor.

The value specified in the CURSOR option must be positive. A negative value leads to unpredictable results.

If this option is omitted, the cursor is positioned at position zero of the screen.

DEFAULT
sets the terminal to use the DEFAULT screen size.

ERASE
specifies that the screen printer buffer or partition is to be erased and the

cursor returned to the upper left corner of the screen. (This option applies only to the 3270, or 8775, and to the 3604 Keyboard Display.)

The first output operation in any transaction, or in a series of pseudoconversational transactions, should always specify ERASE. For transactions attached to 3270 screens or printers, unless explicitly overridden by the DEFAULT or ALTERNATE option, this also ensures that the correct screen size is selected, as defined for the transaction by the SCRNSIZE option in the RDO PROFILE definition.

ERASEAUP
specifies that all unprotected character locations in the partition or the entire screen are to be erased. (This option applies only to the 3270 and 8775.)

FORMFEED
specifies that a new page is required. For 3270 printers and displays, the FORMFEED character is positioned at the start of the buffer. The application program must thus ensure that this buffer position is not overwritten by map or text data. It is ignored if the target terminal does not support FORMFEED (that is, the RDO TYPETERM option FORMFEED was not used).

FREEKB
specifies that the 3270 keyboard is to be unlocked. If FREEKB is omitted, the keyboard remains locked.

Note that the keyboard lock status is maintained separately for each partition on a terminal that supports partitions.

FRSET
specifies that the modified data tags (MDTs) of all fields currently in the 3270 (or partition) buffer are to be reset to the unmodified condition (that is, field reset).

This allows the ATTRB operand of DFHMDF for the next requested map to control the final status of fields written or rewritten in response to a BMS command, if no other attribute information has been written in the symbolic map.

HONEOM
specifies that the default printer line length is to be used. This length should be the same as that specified using the RDO TYPETERM options PAGESIZE or ALTPAGE.

LAST
specifies that this is the last output operation for a transaction and, therefore, the end of a bracket. This option applies to logical units only.

LDC(*name*)
specifies a 2-character mnemonic to be used to determine the logical device code (LDC) to be transmitted in the FMH to the logical unit. The mnemonic identifies an LDC entry defined by a DFHTCT TYPE=LDC macro. When an LDC is specified, BMS uses the device type, the page size, and the page status associated with the LDC mnemonic to format the message. These values are taken from the extended local LDC table for the LU, if it has one. If the LU has only a local (unextended) LDC table, the values are taken from the system LDC table. The numeric value of the LDC is obtained from the local LDC table, unless this is an unextended table and the value is not specified, in which case it is taken from the system table.

L40, L64, or L80
specifies the line length for a 3270 printer; a carrier return and line feed are

forced after 40, 64, or 80 characters have been printed on a line. Unacceptable results are likely if this differs from the page width specified by the RDO TYPETERM options PAGESIZE or ALTPAGE.

When using the options, refer to DFHMDI options, CTRL for a description of the option priority.

MSR(*data-value***)**
specifies the 4-byte data value that controls the 10/63 magnetic stripe reader attached to an 8775 or 3643 terminal. A set of constants is provided in DFHMSRCA to assist in setting this 4-byte area. See "Magnetic slot reader (MSR) control value constants, DFHMSRCA" on page 852 for a complete list. This option is ignored if the RDO TYPETERM option MSRCONTROL was not used.

OUTPARTN(*name***)**
specifies the name (1–2 characters) of the partition to which data is to be sent. This option is ignored if the terminal does not support partitions, or if there is no application partition set associated with the terminal. If there is an application partition set, and the OUTPARTN option is omitted, data is sent to the partition named by the PARTN operand of the DFHMSD (see "DFHMSD" on page 877) or the DFHMDI (see "DFHMDI" on page 868) map definition macros. If maps are not used, or if there is no PARTN operand, the output is sent to the first partition in the partition set.

PAGING
specifies that the output data is not to be sent immediately to the terminal, but is to be placed in temporary storage and displayed in response to paging commands entered by the terminal operator.

If PAGING is specified with a REQID prefix that is used for temporary storage queues that are defined as recoverable, CICS provides message recovery for logical messages if the task has reached a syncpoint.

PRINT
specifies that a print operation is to be started at a 3270 printer or at a 3275 with the printer adapter feature, or that data on an LUTYPE2 (3274/76 or 3790) is to be printed on a printer allocated by the controller. If this option is omitted, the data is sent to the printer buffer but is not printed.

REQID(*name***)**
specifies a 2-character prefix to be used as part of a temporary storage identifier for CICS message recovery. Only one prefix can be specified for each logical message. The default prefix is **.

BMS message recovery is provided for a logical message only if the PAGING option is specified in the BMS SEND commands, and if the syncpoint has been reached.

SET(*ptr-ref***)**
specifies the pointer to be set to the address of the output data.

The SET option specifies that completed pages are to be returned to the application program. The pointer is set to the address of a list of completed pages.

If TIOAPFX=YES is specified in the map definition, the pointer returned contains the address of the TIOA prefix. The user data starts at offset X'0C' from the start of the TIOA prefix.

TERMINAL
specifies that the output data is to be sent to the terminal that originated the transaction.

WAIT
specifies that control should not be returned to the application program until the output operation has been completed.

If WAIT is not specified, control returns to the application program when the output operation has started. A subsequent input or output command (terminal control, BMS, or batch data interchange) causes the application program to wait until the previous command has been completed.

Conditions

57 IGREQCD
occurs when an attempt is made to execute a SEND CONTROL command after a SIGNAL data-flow control command with a request change direction (RCD) code has been received from an LUTYPE4 logical unit.

Default action: terminate the task abnormally.

39 IGREQID
occurs if the prefix specified in the REQID option is different from that established by a previous REQID option, or by default for this logical message—REQID (**).

Default action: terminate the task abnormally.

41 INVLDC
occurs if the specified LDC mnemonic is not included in the LDC list for the logical unit.

Default action: terminate the task abnormally.

65 INVPARTN
occurs if the specified partition is not defined in the partition set associated with the application program.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application attempted to send on its function-shipping session (its principal facility).

also occurs (RESP2 not set) in the following situation:

- Control information is output to the same partition or LDC as mapped data while a BMS logical message is active. If neither partitions nor LDCs are in use, control information is output to the same device as mapped data.

Default action: terminate the task abnormally.

32 RETPAGE
occurs if the SET option is specified and a completed page is ready for return to the application program.

Default action: return control to the application program at the point immediately following the BMS SEND command.

35 TSIOERR
occurs if there is an irrecoverable temporary storage input/output error.

Default action: terminate the task abnormally.

03 WRBRK
occurs if the command is interrupted by the terminal operator pressing the ATTN key. It applies only to the 2741 Communication Terminal, and only if write break is supported for CICS.

Default action: ignore the condition.

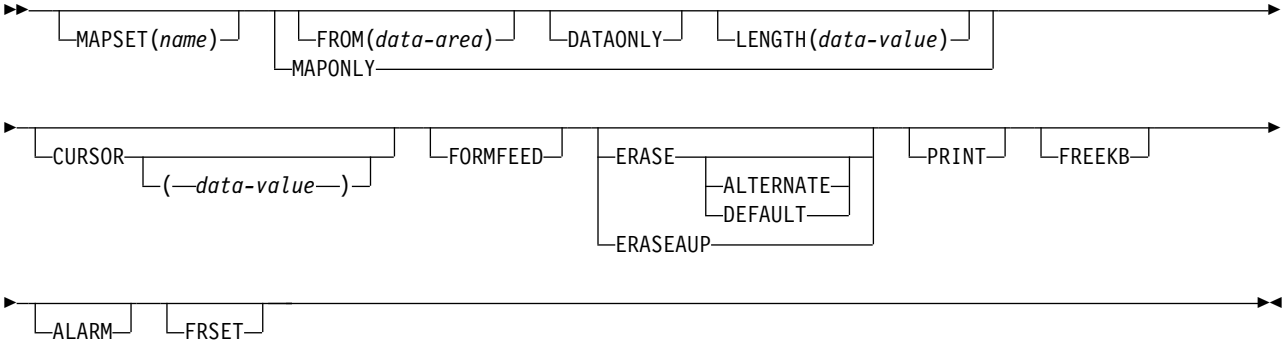
SEND MAP

Send mapped output data to a terminal. The keywords are separated into those supported by minimum, standard, and full BMS. For further information about BMS, see Basic mapping support.

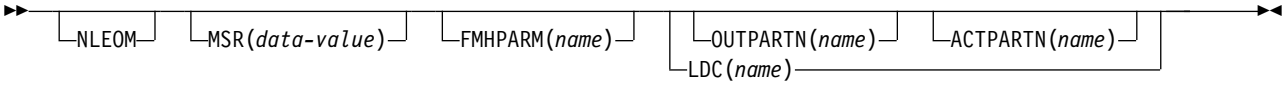
SEND MAP



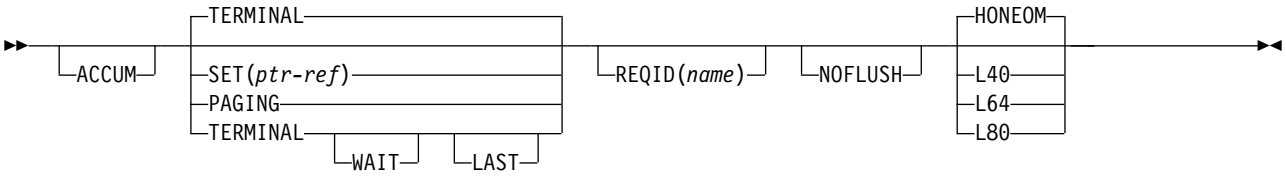
SEND MAP Minimum BMS



SEND MAP Standard BMS



SEND MAP Full BMS



Conditions: IGREQCD, IGREQID, INVLDC, INVMPSZ, INVPARTN, INVREQ, OVERFLOW, RETPAGE, TSIOERR, WRBRK

Description

SEND MAP sends output data to a terminal.

When using the SEND MAP command with any of the ALARM, FREEKB, FRSET, HONEOM, L40, L64, L80, or PRINT options, see “DFHMDI” on page 868 for a description of the option priority.

See BMS macros for map definition.

Options

ACCUM
specifies that this command is one of a number of commands that are used to build a logical message. The logical message is completed by a SEND PAGE command, or deleted by a PURGE MESSAGE command.

ACTPARTN(*name***)**
specifies the name (1–2 characters) of the partition to be activated. Activating a partition moves the cursor into the specified partition, and unlocks the keyboard for the specified partition.

This option is ignored if the target terminal does not support partitions, or if there is no application partition set.

ALARM
specifies that the 3270 audible alarm feature is to be activated. For logical units supporting FMHs (except interactive and batch logical units), ALARM instructs BMS to set the alarm flag in the FMH.

When using the ALARM option, refer to “DFHMDI” on page 868 for a description of the option priority.

ALTERNATE
sets the terminal to use the ALTERNATE screen size.

CURSOR(*data-value***)**
specifies the location to which the 3270 or 3604 cursor is to be returned upon completion of a SEND MAP command.

The data value must be a halfword binary value that specifies the cursor position relative to zero; the range of values that can be specified depends on the size of the screen being used. If no data value is specified, symbolic cursor positioning is assumed.

This option overrides any IC option of the ATTRB operand of DFHMDF. If ACCUM is being used, the most recent value of CURSOR specified is used to position the cursor.

The value specified in the CURSOR option must be positive. A negative value leads to unpredictable results.

DATAONLY
specifies that only application program data is to be written. The attribute characters (3270 only) must be specified for each field in the supplied data. If the attribute byte in the user-supplied data is set to X'00', the attribute byte on the screen is unchanged. Any default data or attributes from the map are ignored.

DEFAULT
sets the terminal to use the DEFAULT screen size.

ERASE
specifies that the screen printer buffer or partition is to be erased and the cursor returned to the upper left corner of the screen. (This option applies only to the 3270, or 8775, and to the 3604 Keyboard Display.)

The first output operation in any transaction, or in a series of pseudoconversational transactions, should always specify ERASE. For transactions attached to 3270 screens or printers, unless explicitly overridden by the DEFAULT or ALTERNATE option, this also ensures that the correct screen size is selected, as defined for the transaction by the SCRNSIZE option in the RDO PROFILE definition.

ERASEAUP
specifies that before this page of output is displayed, all unprotected character locations in the partition or the entire screen are to be erased. (This option applies only to the 3270 and 8775.)

FMHPARM(*name***)**
specifies the name (1–8 characters) of the outboard map to be used. (This option applies only to 3650 logical units with outboard formatting.)

FORMFEED
specifies that a new page is required. For 3270 printers and displays, the FORMFEED character is positioned at the start of the buffer. The application program must thus ensure that this buffer position is not overwritten by map or text data. It is ignored if the target terminal does not support FORMFEED (that is, the RDO TYPETERM option FORMFEED was not used).

FREEKB
specifies that the 3270 keyboard should be unlocked after the data is written. If FREEKB is omitted, the keyboard remains locked.

Note that the keyboard lock status is maintained separately for each partition on a terminal that supports partitions.

When using the FREEKB option, refer to DFHMDI options, CTRL for a description of the option priority.

FROM(*data-area***)**
specifies the data area containing the data to be processed. If this field is not specified, the name defaults to the name of the map suffixed with an O. This includes the 12-byte prefix generated by the TIOAPFX=YES option on the DFHMDI and DFHMSD BMS map definitions (see “DFHMDI” on page 868 and specifying NODDS in the BMS operand).

FRSET
specifies that the modified data tags (MDTs) of all fields currently in the 3270 (or partition) buffer are to be reset to the unmodified condition (that is, field reset) before any map data is written to the buffer.

This allows the ATTRB operand of DFHMDF for the requested map to control the final status of fields written or rewritten in response to a BMS command, if no other attribute information has been written in the symbolic map.

When using the FRSET option refer to “DFHMDI” on page 868 for a description of the option priority.

HONEOM
specifies that the default printer line length is to be used. This length should be the same as that specified using the RDO TYPETERM options PAGESIZE or ALTPAGE, and the same as the printer platen width; otherwise the data may not format correctly.

When using the HONEOM option, refer to “DFHMDI” on page 868 for a description of the option priority.

LAST
specifies that this is the last output operation for a transaction and, therefore, the end of a bracket. This option applies to logical units only.

LDC(*name***)**
specifies a 2-character mnemonic to be used to determine the logical device code (LDC) to be transmitted in the FMH to the logical unit. The mnemonic identifies an LDC entry defined by a DFHTCT TYPE=LDC macro.

When an LDC is specified, BMS uses the device type, the page size, and the page status associated with the LDC mnemonic to format the message. These values are taken from the extended local LDC table for the logical unit, if it has one. If the logical unit has only a local (unextended) LDC table, the values are taken from the system LDC table. The numeric value of the LDC is obtained from the local LDC table, unless this is an unextended table and the value is not specified, in which case it is taken from the system table.

If the LDC option is omitted, the LDC mnemonic specified in the DFHMSD macro is used; see "DFHMSD" on page 877. If the LDC option has also been omitted from the DFHMSD macro, the action depends on the type of logical unit, as follows:

3601 logical unit
The first entry in the local or extended local LDC table is used, if there is one. If a default cannot be obtained in this way, a null LDC numeric value (X'00') is used. The page size used is the value that is specified in the RDO TYPETERM options PAGESIZE or ALTPAGE, or (1,40) if such a value is not specified.

LUTYPE4 logical unit, batch logical unit, or batch data interchange logical unit
The local LDC table is not used to supply a default LDC; instead, the message is directed to the logical unit console (that is, to any medium that the logical unit elects to receive such messages). For a batch data interchange logical unit, this does not imply sending an LDC in an FMH. The page size is obtained in the manner described for the 3601 logical unit.

LENGTH(*data-value***)**
specifies the length of the data to be formatted as a halfword binary value.

If the data area sending the map is longer than the data to be mapped, LENGTH should be specified. This should include the length of the 12-byte prefix generated by the TIOAPFX=YES option on the DFHMDI and DFHMSD BMS map definitions (see "DFHMDI" on page 868 and specifying NODDS in the BMS operand). For a description of a safe upper limit, see "LENGTH options in CICS commands" on page 10.

L40, L64, or L80
specifies the line length for a 3270 printer; a carrier return and line feed are forced after 40, 64, or 80 characters have been printed on a line. Unacceptable results are likely if this differs from the page width specified by the RDO TYPETERM options PAGESIZE or ALTPAGE.

When using the options, refer to "DFHMDI" on page 868 for a description of the option priority.

MAP(*name***)**
specifies the name (1–7 characters) of the map to be used.

MAPONLY
specifies that only default data from the map is to be written.

MAPSET(*name*)
specifies the unsuffixed name (1–7 characters) of the mapset to be used. The mapset must reside in the CICS program library. The mapset can be defined either by using RDO or by program autoinstall when the mapset is first used. If this option is not specified, the name given in the MAP option is assumed to be that of the mapset.

The number of maps per mapset is limited to a maximum of 9 998.

MSR(*data-value*)
specifies the 4-byte data value that controls the 10/63 magnetic stripe reader attached to an 8775 or 3643 terminal. A set of constants is provided in DFHMSRCA to assist in setting this 4-byte area. See "Magnetic slot reader (MSR) control value constants, DFHMSRCA" on page 852 for a complete list. This option is ignored if the RDO TYPETERM option MSRCONTROL was not used.

NLEOM
specifies that data for a 3270 printer or a 3275 display with the printer adapter feature should be built with blanks and new-line (NL) characters, and that an end-of-message (EM) character should be placed at the end of the data. As the data is printed, each NL character causes printing to continue on the next line, and the EM character terminates printing.

This option must be specified in the first SEND MAP command used to build a logical message. The option is ignored if the device receiving the message (direct or routed) is not one of those mentioned above.

If this option is used, buffer updating and attribute modification of fields previously written into the buffer are not allowed. CICS includes the ERASE option with every write to the terminal.

The NL character occupies a buffer position. A number of buffer positions, equivalent to the value of the RDO TYPETERM options PAGESIZE or ALTPAGE, for that terminal, is unavailable for data. This may cause data to wrap around in the buffer; if this occurs, the PAGESIZE or ALTPAGE value must be reduced.

The NLEOM option overrides the ALARM option if the latter is present.

NOFLUSH
specifies that CICS does not clear pages on completion but returns control to the program (having set the OVERFLOW condition in EIBRESP).

OUTPARTN(*name*)
specifies the name (1–2 characters) of the partition to which data is to be sent. This option is ignored if the terminal does not support partitions, or if there is no application partition set associated with the terminal. If there is an application partition set, and the OUTPARTN option is omitted, data is sent to the partition named by the PARTN operand of the DFHMSD or DFHMDI map definitions. If maps are not used, or if there is no PARTN operand, the output is sent to the first partition in the partition set.

PAGING
specifies that the output data is not to be sent immediately to the terminal, but is to be placed in temporary storage and displayed in response to paging commands entered by the terminal operator.

If PAGING is specified with a REQID prefix that is used for temporary storage queues that are defined as recoverable, CICS provides message recovery for logical messages if the task has reached a syncpoint.

PRINT
specifies that a print operation is to be started at a 3270 printer or at a 3275 with the printer adapter feature, or that data on an LUTYPE2 (3274/76 or 3790) is to be printed on a printer allocated by the controller. If this option is omitted, the data is sent to the printer buffer but is not printed.

When using the PRINT option, refer to "DFHMDI" on page 868 for a description of the option priority.

REQID(*name***)**
specifies a 2-character prefix to be used as part of a temporary storage identifier for CICS message recovery. Only one prefix can be specified for each logical message. The default prefix is **.

BMS message recovery is provided for a logical message only if the PAGING option is specified in the BMS SEND commands and if the syncpoint has been reached.

SET(*ptr-ref***)**
specifies the pointer to be set to the address of the input or output data.

The SET option specifies that completed pages are to be returned to the application program. The pointer is set to the address of a list of completed pages.

The application program regains control either immediately following the SEND MAP command (if the current page is not yet completed), or at the label specified in a HANDLE CONDITION RETPAGE command, if the page has been completed.

If TIOAPFX=YES is specified in the map definition, the pointer returned contains the address of the TIOA prefix. The user data starts at offset X'0C' from the start of the TIOA prefix.

TERMINAL
specifies that the output data is to be sent to the terminal that originated the transaction.

WAIT
specifies that control should not be returned to the application program until the output operation has been completed.

If WAIT is not specified, control returns to the application program when the output operation has started. A subsequent input or output command (terminal control, BMS, or batch data interchange) causes the application program to wait until the previous command has been completed.

Conditions

Some of the following conditions may occur in combination. If more than one occurs, only the first is passed to the application program.

57 IGREQCD
occurs when an attempt is made to execute a SEND MAP command after a SIGNAL data-flow control command with a request change direction (RCD) code has been received from an LUTYPE4 logical unit.

Default action: terminate the task abnormally.

39 IGREQID
occurs if the prefix specified in the REQID option is different from that established by a previous REQID option, or by default for this logical message -REQID (**).

Default action: terminate the task abnormally.

41 INVLDC
occurs if the specified LDC mnemonic is not included in the LDC list for the logical unit.

Default action: terminate the task abnormally.

38 INVMPSZ
occurs if the specified map is too wide for the terminal, or if a HANDLE CONDITION OVERFLOW command is active and the specified map is too long for the terminal.

Default action: terminate the task abnormally.

65 INVPARTN
occurs if the specified partition is not defined in the partition set associated with the application program.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 Command not allowed for a distributed program link server program.

also occurs (RESP2 not set) in any of the following situations:

- Text data is output to the same partition or LDC as mapped data while a BMS logical message is active. If neither partitions nor LDCs are in use, text data is output to the same logical message as mapped data.
- A separate SEND MAP command with the ACCUM option is issued to the terminal that originated the transaction while a routed logical message is being built.
- A SEND MAP command is issued for a map without field specifications by specifying the FROM option without the DATAONLY option.
- During overflow processing, data is sent to a different LDC from the LDC that caused page overflow.
- Partitions are in use, the OUTPARTN option has not been coded on the SEND MAP command, but the PARTN operand has been coded in the mapset definition. If the condition arises, it suggests that different versions of the mapset have different PARTN values, and that the suffix deduced for the partition is not the same as the suffix of the loaded mapset.
- A SEND MAP command with the DATAONLY option is issued with a data area, supplied by the user, that resides above the 16MB line. But the length of this data area is not longer than the TIOA prefix.

Default action: terminate the task abnormally.

40 OVERFLOW
occurs if the mapped data does not fit on the current page. This condition is only raised if a HANDLE CONDITION OVERFLOW command is active.

Default action: ignore the condition.

32 RETPAGE
occurs if the SET option is specified and a completed page is ready for return to the application program.

Default action: return control to the application program at the point immediately following the BMS SEND MAP command.

35 TSIOERR
occurs if there is an irrecoverable temporary storage input/output error.

Default action: terminate the task abnormally.

03 WRBRK
occurs if a SEND MAP command is interrupted by the terminal operator pressing the ATTN key. It applies only to the 2741 Communication Terminal, and only if write break is supported for CICS.

Default action: ignore the condition.

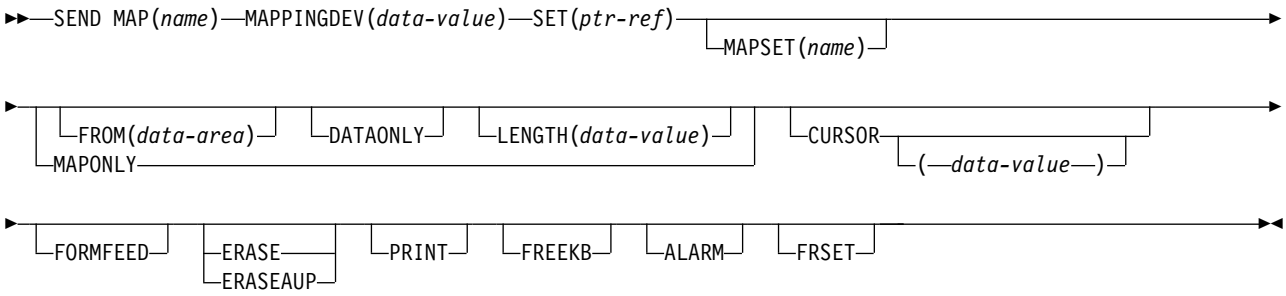
SEND MAP MAPPINGDEV

Create mapped output data to be sent to a terminal described by MAPPINGDEV at some later time.

For further information about BMS, see Basic mapping support.

Minimum BMS:

SEND MAP MAPPINGDEV



Conditions: INVMPSZ, INVREQ

Description

SEND MAP MAPPINGDEV creates mapped output data to be sent to a terminal that is not the principal facility of the transaction. The terminal characteristics to be used are defined by MAPPINGDEV.

The mapped data is not transmitted but is returned to the application in a buffer defined by the SET option.

Options

ALARM
specifies that the 3270 audible alarm feature is to be activated. For logical units supporting FMHs (except interactive and batch logical units), ALARM instructs BMS to set the alarm flag in the FMH.

When using the ALARM option, refer to DFHMDI options, CTRL for a description of the option priority.

CURSOR(*data-value***)**
specifies the location to which the 3270 cursor is to be returned upon completion of a SEND MAP MAPPINGDEV command.

The data value must be a halfword binary value that specifies the cursor position relative to zero; the range of values that can be specified depends on the size of the screen being used. If no data value is specified, symbolic cursor positioning is assumed.

This option overrides any IC option of the ATTRB operand of DFHMDF.

The value specified in the CURSOR option must be positive. A negative value leads to unpredictable results.

DATAONLY
specifies that only application program data is to be written. The attribute characters (3270 only) must be specified for each field in the supplied data. If the attribute byte in the user-supplied data is set to X'00', the attribute byte on the screen is unchanged. Any default data or attributes from the map are ignored.

ERASE
specifies that the screen printer buffer is to be erased and the cursor returned to the upper left corner of the screen. (This option applies only to the 3270, or 8775)

The first output operation in any transaction, or in a series of pseudoconversational transactions, should always specify ERASE. For transactions attached to 3270 screens or printers, this also ensures that the correct screen size is selected, as defined for the transaction by the SCRNSIZE option in the RDO PROFILE definition.

ERASEAUP
specifies that before this page of output is displayed, all unprotected character locations in the entire screen are to be erased. (This option applies only to the 3270 and 8775.)

FORMFEED
specifies that a new page is required. For 3270 printers and displays, the FORMFEED character is positioned at the start of the buffer. The application program must thus ensure that this buffer position is not overwritten by map or text data. It is ignored if the target terminal does not support FORMFEED (that is, the RDO TYPETERM option FORMFEED was not used, or the terminal control table TYPE=TERMINAL does not specify FF=YES).

FREEKB
specifies that the 3270 keyboard should be unlocked after the data is written. If FREEKB is omitted, the keyboard remains locked.

When using the FREEKB option, refer to CTRL DFHMDI options, CTRL for a description of the option priority.

FROM(*data-area***)**
specifies the data area containing the data to be processed. If this field is not specified, the name defaults to the name of the map suffixed with an O. This includes the 12-byte prefix generated by the TIOAPFX=YES option on the DFHMDI and DFHMSD BMS map definitions (see DFHMDI operands, TERM and Specifying NODDS in the BMS operand).

FRSET
specifies that the modified data tags (MDTs) of all fields currently in the 3270 buffer are to be reset to the unmodified condition (that is, field reset) before any map data is written to the buffer.

This allows the ATTRB operand of DFHMDF for the requested map to control the final status of fields written or rewritten in response to a BMS command, if no other attribute information has been written in the symbolic map.

When using the FRSET option refer to DFHMDI options, CTRL for a description of the option priority.

LENGTH(*data-value***)**
specifies the length of the data to be formatted as a halfword binary value.

If the data area sending the map is longer than the data to be mapped, LENGTH should be specified. This should include the length of the 12-byte prefix generated by the TIOAPFX=YES option on the DFHMDI and DFHMSD BMS map definitions (see DFHMDI operands, TERM and Specifying NODDS in the BMS operand). For a description of a safe upper limit, see "LENGTH options in CICS commands" on page 10.

MAP(*name***)**
specifies the name (1–7 characters) of the map to be used.

MAPPINGDEV(*data-value***)**
specifies the name of a 3270 terminal whose BMS characteristics match those of the terminal to which the data will eventually be sent using a SEND TEXT MAPPED command or a terminal control SEND or CONVERSE.

MAPONLY
specifies that only default data from the map is to be written.

MAPSET(*name***)**
specifies the unsuffixed name (1–7 characters) of the mapset to be used. The mapset must reside in the CICS program library. The mapset can be defined either by using RDO or by program autoinstall when the mapset is first used. If this option is not specified, the name given in the MAP option is assumed to be that of the mapset.

The number of maps per mapset is limited to a maximum of 9 998.

PRINT
specifies that a print operation is to be started at a 3270 printer or at a 3275 with the printer adapter feature, or that data on an LUTYPE2 (3274/76 or 3790) is to be printed on a printer allocated by the controller. If this option is omitted, the data is sent to the printer buffer but is not printed.

When using the PRINT option, refer to DFHMDI options, CTRL for a description of the option priority.

SET(*ptr-ref***)**
specifies the pointer to be set to the address of the mapped data.

The storage area containing the mapped data has the same format as the page buffer returned when using the SET option in the full BMS SEND command. For more guidance about using the SET option, see The MAPPINGDEV facility.

Conditions

Some of the following conditions may occur in combination. If more than one occurs, only the first is passed to the application program.

38 INVMPSZ
occurs if the specified map is too wide for the terminal specified by MAPPINGDEV or if a HANDLE CONDITION OVERFLOW command is active and the specified map is too long for the terminal specified by MAPPINGDEV.

Default action: terminate the task abnormally.

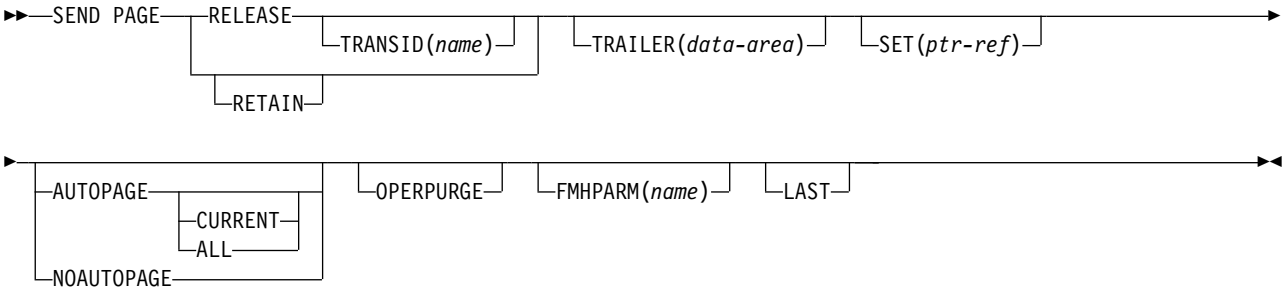
16 INVREQ
occurs if the terminal specified by MAPPINGDEV does not exist, does not support BMS, or is not a 3270 printer or display.

Default action: terminate the task abnormally.

SEND PAGE

Send last page of data.

SEND PAGE



Conditions: IGREQCD, INVREQ, RETPAGE, TSIOERR, WRBRK

Description

SEND PAGE completes a BMS logical message. It causes BMS to generate a device-dependent data stream for the last (perhaps the only) page of data. Typically, this last page is only partially full.SEND PAGE is supplied only by full BMS. For further information about BMS, see Basic mapping support.

Options can be included to specify how much control the terminal operator should have over the disposition of the logical message (AUTOPAGE, NOAUTOPAGE, and OPERPURGE), to determine whether control should return to the application program after transmission of the logical message (RELEASE or RETAIN), to add trailer data to a text logical message (TRAILER), and to return the device-dependent data stream for the last page of a logical message to the application program (SET). If this is a paging message, the last page of the logical

message is transmitted to temporary storage and the terminal operator paging transaction is initiated. If it is a terminal logical message, the last page is transmitted to the terminal.

This is supported by full BMS only.

Options

ALL
specifies that if the ATTN key on a 2741 is pressed while a BMS logical message is being sent to the terminal, and the WRBRK condition is not active, transmission of the current page is to cease and no additional pages are to be transmitted. The logical message is deleted.

AUTOPAGE
specifies that each page of a BMS logical message is to be sent to the terminal as soon as it is available. If paging on request is specified for the terminal by the RDO TYPETERM option AUTOPAGE(NO), AUTOPAGE overrides it for this logical message.

AUTOPAGE is assumed for 3270 printers; it does not apply to 3270 display terminals. If neither AUTOPAGE nor NOAUTOPAGE is specified, the terminal has the paging status specified for it using the RDO TYPETERM option AUTOPAGE.

CURRENT
specifies that if the ATTN key on a 2741 is pressed while a BMS logical message is being sent to the terminal, and the WRBRK condition is not active, transmission of the current page is to cease and transmission of the next page (if any) is to begin.

FMHPARM(*name***)**
specifies the name (1–8 characters) of the outboard map to be used. This option applies only to 3650 logical units with outboard formatting.

LAST
specifies that this is the last output operation for a transaction and, therefore, the end of a bracket. If RELEASE is specified, LAST is assumed unless the SEND PAGE command is terminating a routing operation. This option applies to logical units only.

NOAUTOPAGE
specifies that pages of a BMS logical message are to be sent one at a time to the terminal. BMS sends the first page to the terminal when the terminal becomes available or on request of the terminal operator. Subsequent pages are sent to the terminal in response to requests from the terminal operator.

If automatic paging is specified for the terminal by the RDO TYPETERM option AUTOPAGE(YES), NOAUTOPAGE overrides it for this logical message. For logical units, NOAUTOPAGE applies to all pages for all LDCs in the logical message.NOAUTOPAGE does not apply to 3270 printers.

OPERPURGE
specifies that CICS is to delete the BMS logical message only when the terminal operator requests deletion. If the option is omitted, CICS deletes the message if the operator enters data that is not a paging command.

RELEASE
specifies that, after the SEND PAGE command, control is to be returned to CICS.

RETAIN
specifies that after the SEND PAGE command, control is returned to the application program when the operator has finished displaying the pages.

SET(*ptr-ref***)**
specifies the pointer to be set to the address of the output data.

The SET option specifies that the last or only page is returned to the application program. The pointer is set to the address of the current page. A list of addresses is created and, if the ROUTE command is in operation, there is an address entry for each device. If the ROUTE command is not in operation, the list contains only the one entry.

The application program regains control either immediately following the SEND PAGE command (if the current page is not yet completed), or at the label specified in a HANDLE CONDITION RETPAGE command if the page has been completed.

If TIOAPFX=YES is specified in the map definition, the pointer returned contains the address of the TIOA prefix. The user data starts at offset X'0C' from the start of the TIOA prefix.

TRAILER(*data-area***)**
specifies the text data area that contains trailer data to be placed at the bottom of the last page only. The format of the trailer is:

2 bytes
Binary length of the data (n)

2 bytes
Binary zero

n bytes
Data.

See The SEND PAGE command for more information.

TRANSID(*name***)**
specifies the transaction identifier (1–4 alphanumeric characters) to be used with the next input message from the terminal the task is attached to. The identifier must have been defined to CICS via a RDO TRANSACTION resource definition. TRANSID is valid only if SEND PAGE RELEASE is specified.

If this option is specified in a program that is not at the highest logical level, the specified transaction identifier is used only if a new transaction identifier is not provided in another SEND PAGE command (or in a RETURN program control command) issued in a program at a higher logical level.

Conditions

57 IGREQCD
occurs when an attempt is made to execute a SEND PAGE command after a SIGNAL data-flow control command with a request change direction (RCD) code has been received from an LUTYPE4 logical unit.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 Command not allowed for a distributed program link server program.

also occurs (RESP2 not set) in any of the following situations:

- The disposition (TERMINAL, PAGING, or SET) of a BMS logical message is changed before its completion by the SEND PAGE command.
- Text data is output to the same partition or LDC as mapped data while a BMS logical message is active. If neither partitions nor LDCs are in use, text data is output to the same logical message as mapped data.
- The TRAILER option is specified when terminating a logical message built with SEND MAP commands only.
- During overflow processing data is sent to a different LDC from the LDC that caused page overflow.
- The length of the trailer is negative.

Default action: terminate the task abnormally.

32 RETPAGE
occurs if the SET option is specified and the last or only completed page is ready for return to the application program.

Default action: return control to the application program at the point immediately following the BMS SEND PAGE command.

35 TSIOERR
occurs if there is an unrecoverable temporary storage input/output error.

Default action: terminate the task abnormally.

03 WRBRK
occurs if the SEND PAGE command is interrupted by the terminal operator pressing the ATTN key. It applies only to the 2741 Communication Terminal, and only if write break is supported for CICS.

Default action: ignore the condition.

SEND PARTNSET

Associates the partition set specified by the PARTNSET option with the application program.

SEND PARTNSET



Conditions: INVPARTNSET, INVREQ

Description

SEND PARTNSET associates the partition set specified by the PARTNSET option with the application program. If the partition set name is omitted, the terminal is reset to the base (unpartitioned) state. This command is available on standard and full BMS only. For further information about BMS, see Basic mapping support.

Note: A SEND PARTNSET command must not be followed immediately by a RECEIVE command. The two commands must be separated by a SEND MAP, SEND TEXT, or SEND CONTROL command, so that the partition set is sent to the terminal.

Conditions

The following conditions may occur together. If both occur, only the first one is passed to the application program.

64 INVPARTNSET
occurs if the partition set named in the SEND PARTNSET command is not a valid partition set (for example, it may be a mapset).

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 Command not allowed for a distributed program link server program.

also occurs (RESP2 not set) in the following situation:

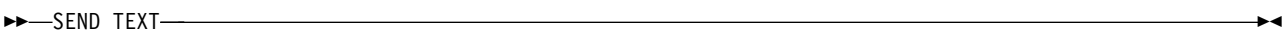
- A SEND PARTNSET command is issued while a logical message is active.

Default action: terminate the task abnormally.

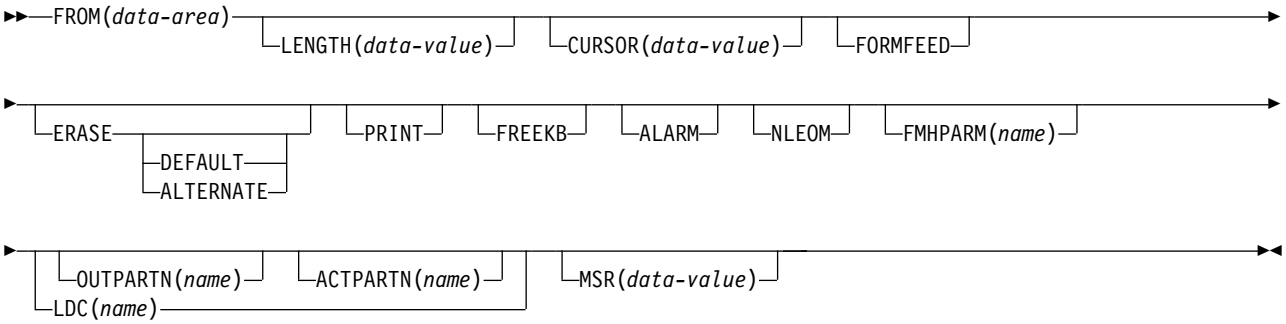
SEND TEXT

Send data without mapping. The keywords are separated into those supported by standard and full BMS. For further information about BMS, see Basic mapping support.

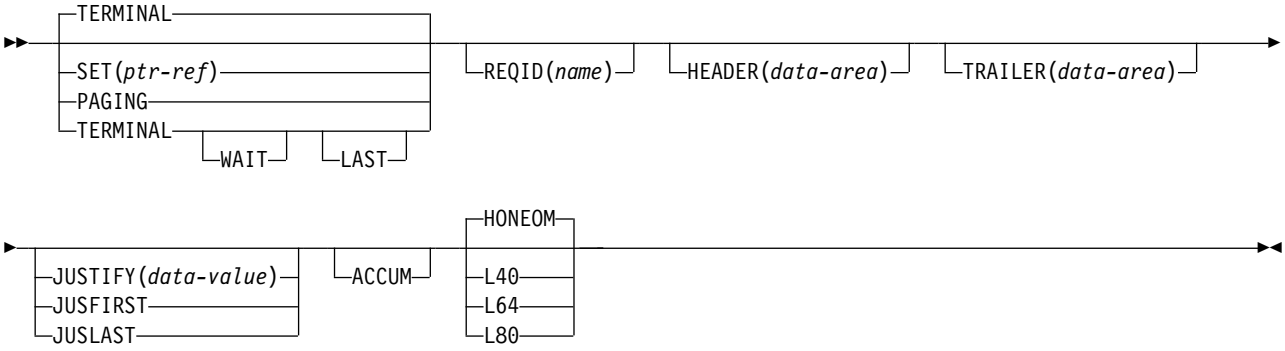
SEND TEXT



SEND TEXT Standard BMS



SEND TEXT Full BMS



Conditions: IGREQCD, IGREQID, INVLDC, INVPARTN, INVREQ, LENGERR, RETPAGE, TSIOERR, WRBRK

Description

SEND TEXT sends text data without mapping. The text is split into lines of the same width as the terminal, such that words are not broken across line boundaries. If the text exceeds a page, it is split into pages that fit on the terminal with application-defined headers and trailers.

When using the SEND TEXT command with any of the ALARM, FREEKB, FRSET, HONEOM, L40, L64, L80, or PRINT options, see DFHMDI options, CTRL for a description of the option priority.

Options

ACCUM
specifies that this command is one of a number of commands that are used to build a logical message. The logical message is completed by a SEND PAGE command, or deleted by a PURGE MESSAGE command.

HEADER, JUSFIRST, JUSLAST, JUSTIFY and TRAILER all imply ACCUM.

ACTPARTN(*name*)
specifies the name (1–2 characters) of the partition to be activated. Activating a partition moves the cursor into the specified partition, and unlocks the keyboard for the specified partition.

This option is ignored if the target terminal does not support partitions, or if there is no application partition set.

ALARM
specifies that the 3270 audible alarm feature is to be activated. For logical units supporting FMHs (except interactive and batch logical units), ALARM instructs BMS to set the alarm flag in the FMH.

ALTERNATE
sets the terminal to use the ALTERNATE screen size.

CURSOR(*data-value*)
specifies the location to which the 3270 or 3604 cursor is to be returned on completion of a SEND TEXT command.

The data value must be a halfword binary value that specifies the cursor position relative to zero; the range of values that can be specified depends on the size of the screen being used.

This option overrides any IC option of the ATTRB operand of DFHMDF. If ACCUM is being used, the most recent value of CURSOR specified is used to position the cursor.

The value specified in the CURSOR option must be positive. A negative value leads to unpredictable results.

DEFAULT
sets the terminal to use the DEFAULT screen size.

ERASE
specifies that the screen printer buffer or partition is to be erased and the cursor returned to the upper left corner of the screen. (This option applies only to the 3270, or 8775, and to the 3604 Keyboard Display.)

The first output operation in any transaction, or in a series of pseudoconversational transactions, should always specify ERASE. For transactions attached to 3270 screens or printers, unless explicitly overridden by the DEFAULT or ALTERNATE option, this also ensures that the correct screen size is selected, as defined for the transaction by the SCRNSIZE option in the RDO PROFILE definition.

FMHPARM(*name***)**
specifies the name (1–8 characters) of the outboard map to be used. (This option applies only to 3650 logical units with outboard formatting.)

FORMFEED
specifies that a new page is required. For 3270 printers and displays, the FORMFEED character is positioned at the start of the buffer. The application program must thus ensure that this buffer position is not overwritten by map or text data. It is ignored if the target terminal does not support FORMFEED (that is, the RDO TYPETERM option FORMFEED was not used).

The FORMFEED option can appear on any SEND TEXT ACCUM command. You need only specify it once within a physical page because it always forces a FORMFEED at the start of the physical page. To force a FORMFEED at the start of a particular SEND TEXT ACCUM command, use the JUSFIRST option instead.

FREEKB
specifies that the 3270 keyboard should be unlocked after the data is written. If FREEKB is omitted, the keyboard remains locked.

When using the FREEKB option, see DFHMDI options, CTRL for a description of the option priority.

Note that the keyboard lock status is maintained separately for each partition on a terminal that supports partitions.

FROM(*data-area***)**
specifies the data area containing the data to be sent.

HEADER(*data-area***)**
specifies the header data to be placed at the beginning of each page of text data. The format of the header is:

2 bytes
Binary length of the data (n).

1 byte Page numbering required or not (blank).

1 byte Reserved field.

n bytes
Data.

See Header and trailer format for more information.

HONEOM
specifies that the default printer line length is to be used. This length should be the same as that specified using the RDO TYPETERM options PAGESIZE or ALTPAGE, and the same as the printer platen width; otherwise the data may not format correctly.

When using the HONEOM option, see DFHMDI options, CTRL for a description of the option priority.

JUSFIRST
specifies that the text data is to be placed at the top of the page. Any partially formatted page from previous requests is considered to be complete. If the HEADER option is specified, the header precedes the data. See also the description of the JUSTIFY option.

JUSLAST
specifies that the text data is to be positioned at the bottom of the page. The page is considered to be complete after the request has been processed. If the TRAILER option is specified, the trailer follows the data. See also the description of the JUSTIFY option.

JUSTIFY(*data-value***)**
specifies the line of the page at which the text data is to be positioned. The data value must be a halfword binary value in the range 1 through 240. Although they may not be specified as constants, the special values -1 and -2 can be supplied dynamically to signify JUSFIRST or JUSLAST, respectively.

LAST
specifies that this is the last output operation for a transaction and, therefore, the end of a bracket. This option applies to logical units only.

LDC(*name***)**
specifies a 2-character mnemonic to be used to determine the logical device code (LDC) to be transmitted in the FMH to the logical unit. The mnemonic identifies an LDC entry defined by a DFHTCT TYPE=LDC macro.

When an LDC is specified, BMS uses the device type, the page size, and the page status associated with the LDC mnemonic to format the message. These values are taken from the extended local LDC table for the logical unit, if it has one. If the logical unit has only a local (unextended) LDC table, the values are taken from the system LDC table. The numeric value of the LDC is obtained from the local LDC table, unless this is an unextended table and the value is not specified, in which case it is taken from the system table.

LENGTH(*data-value***)**
specifies the length of the data to be sent as a halfword binary value. For a description of a safe upper limit, see “LENGTH options in CICS commands” on page 10.

L40, L64, or L80
specifies the line length for a 3270 printer; a carrier return and line feed are forced after 40, 64, or 80 characters have been printed on a line. Unacceptable

results are likely if this differs from the page width specified by the RDO TYPETERM options PAGESIZE or ALTPAGE.

When using the options, see DFHMDI options, CTRL for a description of the option priority.

MSR(*data-value***)**
specifies the 4-byte data value that controls the 10/63 magnetic stripe reader attached to an 8775 or 3643 terminal. A set of constants is provided in DFHMSRCA to assist in setting this 4-byte area. See "Magnetic slot reader (MSR) control value constants, DFHMSRCA" on page 852 for a complete list. This option is ignored if the RDO TYPETERM option MSRCONTROL was not used.

NLEOM
specifies that data for a 3270 printer or a 3275 display with the printer adapter feature should be built with blanks and new-line (NL) characters, and that an end-of-message (EM) character should be placed at the end of the data. As the data is printed, each NL character causes printing to continue on the next line, and the EM character terminates printing.

This option must be specified in the first SEND TEXT command used to build a logical message. The option is ignored if the device receiving the message (direct or routed) is not one of those mentioned above.

If this option is used, buffer updating and attribute modification of fields previously written into the buffer are not allowed. CICS includes the ERASE option with every write to the terminal.

The NL character occupies a buffer position. A number of buffer positions, equivalent to the value of the RDO TYPETERM options PAGESIZE or ALTPAGE for that terminal, is unavailable for data. This may cause data to wrap around in the buffer; if this occurs, the PGESIZE value must be reduced.

The NLEOM option overrides the ALARM option if the latter is present.

OUTPARTN(*name***)**
specifies the name (1–2 characters) of the partition to which data is to be sent. This option is ignored if the terminal does not support partitions, or if there is no application partition set associated with the terminal. If there is an application partition set, and the OUTPARTN option is omitted, data is sent to the partition named by the PARTN operand of the DFHMSD or DFHMDI map definition. If maps are not used, or if there is no PARTN operand, the output is sent to the first partition in the partition set.

PAGING
specifies that the output data is not to be sent immediately to the terminal, but is to be placed in temporary storage and displayed in response to paging commands entered by the terminal operator.

If PAGING is specified with a REQID prefix that is used for temporary storage queues that are defined as recoverable, CICS provides message recovery for logical messages if the task has reached a syncpoint.

PRINT
specifies that a print operation is to be started at a 3270 printer or at a 3275 with the printer adapter feature, or that data on an LUTYPE2 (3274/76 or 3790) is to be printed on a printer allocated by the controller. If this option is omitted, the data is sent to the printer buffer but is not printed.

When using the PRINT option, see DFHMDI options, CTRL for a description of the option priority.

REQID(*name***)**
specifies a 2-character prefix to be used as part of a temporary storage identifier for CICS message recovery. Only one prefix can be specified for each logical message. The default prefix is **.

BMS message recovery is provided for a logical message only if the PAGING option is specified in the BMS SEND commands and if the syncpoint has been reached.

SET(*ptr-ref***)**
specifies the pointer to be set to the address of the data. It specifies that completed pages are to be returned to the application program. The pointer is set to the address of a list of completed pages.

The application program regains control either immediately following the BMS SEND command (if the current page is not yet completed), or at the label specified in a HANDLE CONDITION RETPAGE command if the page has been completed.

If TIOAPFX=YES is specified in the map definition, the pointer returned contains the address of the TIOA prefix. The user data starts at offset X'0C' from the start of the TIOA prefix.

TERMINAL
specifies that data is to be sent to the terminal that originated the transaction.

TRAILER(*data-area***)**
specifies the text data area that contains trailer data to be placed at the bottom of each output page. The format of the trailer is:
2 bytes
Binary length of the data (n)
1 byte Page numbering required or not (blank)
1 byte Reserved field
n bytes
Data

See the Header and trailer format for more information.

WAIT
specifies that control should not be returned to the application program until the output operation has been completed.

If WAIT is not specified, control returns to the application program when the output operation has started. A subsequent input or output command (terminal control, BMS, or batch data interchange) causes the application program to wait until the previous command has been completed.

Conditions

57 IGREQCD
occurs when an attempt is made to execute a SEND TEXT command after a SIGNAL data-flow control command with a request change direction (RCD) code has been received from an LUTYPE4 logical unit.

Default action: terminate the task abnormally.

39 IGREQID
occurs if the prefix specified in the REQID option on a BMS SEND command is different from that established by a previous REQID option, or by default for this logical message—REQID (**).

Default action: terminate the task abnormally.

41 INVLDC
occurs if the specified LDC mnemonic is not included in the LDC list for the logical unit.

Default action: terminate the task abnormally.

65 INVPARTN
occurs if the specified partition is not defined in the partition set associated with the application program.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 Command not allowed for a distributed program link server program.

also occurs (RESP2 not set) in any of the following situations:

- Text data is output to the same partition or LDC as mapped data while a BMS logical message is active. If neither partitions nor LDCs are in use, text data is output to the same logical message as mapped data.
- During overflow processing, data is sent to a different LDC from the LDC that caused page overflow.
- The length of a header on a SEND TEXT command is negative.
- The length of a trailer on a SEND TEXT command is negative.

Default action: terminate the task abnormally.

22 LENGERR
occurs if an out-of-range value is supplied in the LENGTH option.

Default action: terminate the task abnormally.

32 RETPAGE
occurs if the SET option is specified and a completed page is ready for return to the application program.

Default action: return control to the application program at the point immediately following the BMS SEND TEXT command.

35 TSIOERR
occurs if there is an irrecoverable temporary storage input/output error.

Default action: terminate the task abnormally.

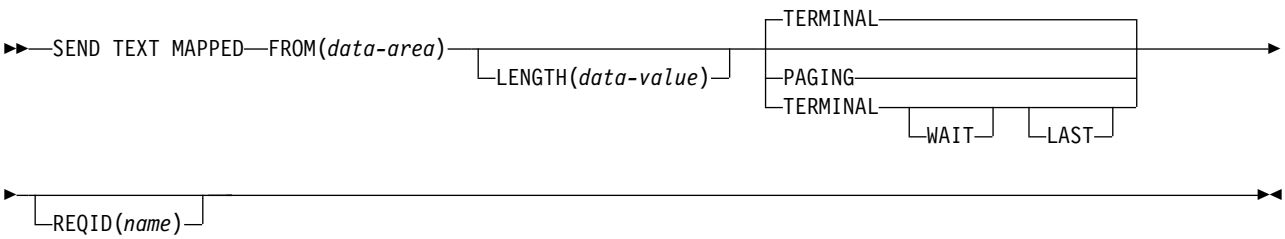
03 WRBRK
occurs if a SEND command is interrupted by the terminal operator pressing the ATTN key. It applies only to the 2741 Communication Terminal, and only if write break is supported for CICS.

Default action: ignore the condition.

SEND TEXT MAPPED

Send data with mapping. Only supplied by full BMS.

SEND TEXT MAPPED



Conditions: IGREQCD, IGREQID, INVREQ, TSIOERR, WRBRK

Description

SEND TEXT MAPPED sends a page of a device-dependent data stream previously built by BMS, and returned to the application program with the SET option.

It must only be used to send data previously generated by a BMS SEND command specifying the SET option. It references a 4-byte page control area (PGA) that BMS placed at the end of the device-dependent data stream.

The length of the device-dependent data stream set in the TIOATDL field of the page buffer returned by the SET option does not include the PGA. The LENGTH option of the SEND TEXT MAPPED command should be set from this TIOATDL, and hence does not include the PGA. However, if the application program copies the page buffer returned by the SET option, it should include the PGA in the copied data.

This command is only supported by full BMS.

Options

FROM(*data-area***)**
specifies the data area containing the data to be sent.

LAST
specifies that this is the last output operation for a transaction and, therefore, the end of a bracket. This option applies to logical units only.

LENGTH(*data-value***)**
specifies the length of the data to be formatted as a halfword binary value. For a description of a safe upper limit, see “LENGTH options in CICS commands” on page 10.

PAGING
specifies that the output data is not to be sent immediately to the terminal, but is to be placed in temporary storage and displayed in response to paging commands entered by the terminal operator.

If PAGING is specified with a REQID prefix that is used for temporary storage queues that are defined as recoverable, CICS provides message recovery for logical messages if the task has reached a syncpoint.

REQID(*name***)**
specifies a 2-character prefix to be used as part of a temporary storage identifier for CICS message recovery. Only one prefix can be specified for each logical message. The default prefix is **.

BMS message recovery is provided for a logical message only if the PAGING option is specified in the BMS SEND commands and if the syncpoint has been reached.

TERMINAL
specifies that input data is to be sent to the terminal that originated the transaction.

WAIT
specifies that control should not be returned to the application program until the output operation has been completed.

If WAIT is not specified, control returns to the application program when the output operation has started. A subsequent input or output command (terminal control, BMS, or batch data interchange) causes the application program to wait intil the previous command has been completed.

Conditions

57 IGREQCD
occurs when an attempt is made to execute a SEND TEXT command after a SIGNAL data-flow control command with a request change direction (RCD) code has been received from an LUTYPE4 logical unit.

Default action: terminate the task abnormally.

39 IGREQID
occurs if the prefix specified in the REQID option on a BMS SEND command is different from that established by a previous REQID option, or by default for this logical message—REQID (**).

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

35 TSIOERR
occurs if there is an irrecoverable temporary storage input/output error.

Default action: terminate the task abnormally.

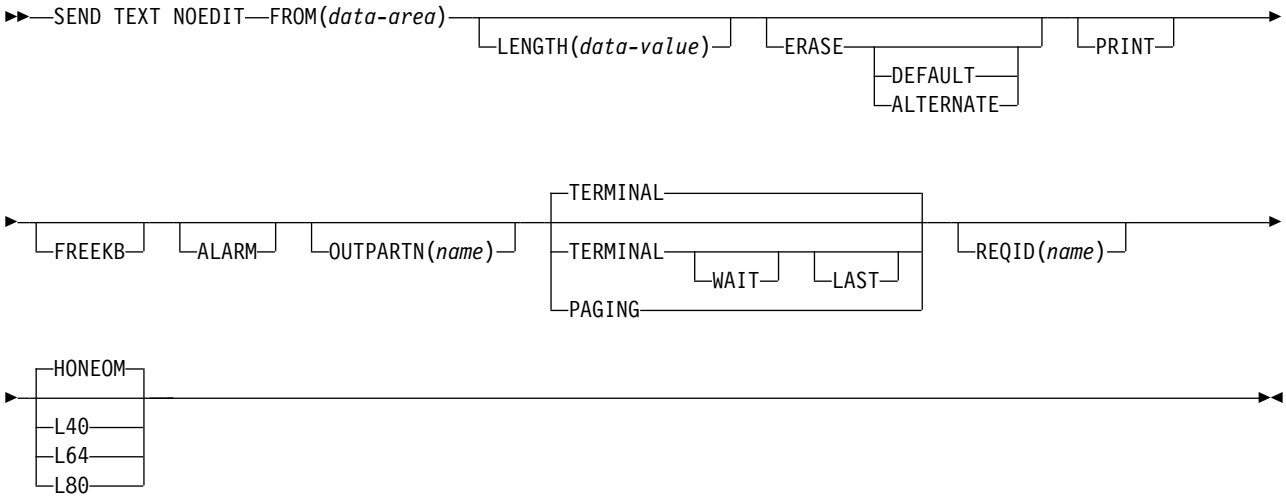
03 WRBRK
occurs if a SEND command is interrupted by the terminal operator pressing the ATTN key. It applies only to the 2741 Communication Terminal, and only if write break is supported for CICS.

Default action: ignore the condition.

SEND TEXT NOEDIT

Send a page. Only supplied by full BMS.

SEND TEXT NOEDIT



Conditions: IGREQCD, IGREQID, INVREQ, INVPARTN, TSIOERR, WRBRK

Description

SEND TEXT NOEDIT sends a page of a device-dependent data stream built by the application program. The data stream cannot contain structured fields. This command differs from a terminal control SEND, because the data stream may be written to temporary storage and interfaced to the terminal operator paging transaction (using the PAGING option). Also the device-dependent data stream can be sent to a partition (using the OUTPARTN option).

If the OUTPARTN option is specified, the data stream is sent to the specified partition. This command is used to output a user-generated data stream. It differs from a terminal control SEND in that data may be output to temporary storage (using the PAGING option), or routed like any other BMS data.

When using the SEND TEXT NOEDIT command with any of the ALARM, FREEKB, FRSET, HONEOM, L40, L64, L80, or PRINT options, see DFHMDI options, CTRL for a description of the option priority.

This command is supported on full BMS only.

Options

ALARM
specifies that the 3270 audible alarm feature is to be activated. For logical units supporting FMHs (except interactive and batch logical units), ALARM instructs BMS to set the alarm flag in the FMH.

ALTERNATE
sets the terminal to use the ALTERNATE screen size.

DEFAULT
sets the terminal to use the DEFAULT screen size.

ERASE
specifies that the screen printer buffer or partition is to be erased and the cursor returned to the upper left corner of the screen. (This option applies only to the 3270, or 8775, and to the 3604 Keyboard Display.)

The first output operation in any transaction, or in a series of pseudoconversational transactions, should always specify ERASE. For transactions attached to 3270 screens or printers, unless explicitly overridden by the DEFAULT or ALTERNATE option, this also ensures that the correct screen size is selected, as defined for the transaction by the SCRNSIZE option in the RDO PROFILE definition.

FREEKB
specifies that the 3270 keyboard should be unlocked after the data is written. If FREEKB is omitted, the keyboard remains locked.

Note that the keyboard lock status is maintained separately for each partition on a terminal that supports partitions.

When using the FREEKB option, refer to DFHMDI options, CTRL for a description of the option priority.

FROM(*data-area***)**
specifies the data area containing the data to be sent.

HONEOM
specifies that the default printer line length is to be used. This length should be the same as that specified using the RDO TYPETERM options PAGESIZE or ALTPAGE, and the same as the printer platen width; otherwise the data may not format correctly.

When using the HONEOM option, refer to DFHMDI options, CTRL for a description of the option priority.

LAST
specifies that this is the last output operation for a transaction and, therefore, the end of a bracket. This option applies to logical units only.

LENGTH(*data-value***)**
specifies the length of the data to be sent as a halfword binary value. For a description of a safe upper limit, see "LENGTH options in CICS commands" on page 10.

L40, L64, or L80
specifies the line length for a 3270 printer; a carrier return and line feed are forced after 40, 64, or 80 characters have been printed on a line. Unacceptable results are likely if this differs from the page width specified by the RDO TYPETERM options PAGESIZE or ALTPAGE.

When using the options, refer to CTRL DFHMDI options, CTRL for a description of the option priority.

OUTPARTN(*name***)**
specifies the name (1–2 characters) of the partition to which data is to be sent. This option is ignored if the terminal does not support partitions, or if there is no application partition set associated with the terminal. If there is an application partition set, and the OUTPARTN option is omitted, data is sent to

the partition named by the PARTN operand of the DFHMSD or DFHMDI map definition. If maps are not used, or if there is no PARTN operand, the output is sent to the first partition in the partition set.

PAGING
specifies that the output data is not to be sent immediately to the terminal, but is to be placed in temporary storage and displayed in response to paging commands entered by the terminal operator.

If PAGING is specified with a REQID prefix that is used for temporary storage queues that are defined as recoverable, CICS provides message recovery for logical messages if the task has reached a syncpoint.

PRINT
specifies that a print operation is to be started at a 3270 printer or at a 3275 with the printer adapter feature, or that data on an LUTYPE2 (3274/76 or 3790) is to be printed on a printer allocated by the controller. If this option is omitted, the data is sent to the printer buffer but is not printed.

When using the PRINT option, refer to DFHMDI options, CTRL for a description of the option priority.

REQID(*name***)**
specifies a 2-character prefix to be used as part of a temporary storage identifier for CICS message recovery. Only one prefix can be specified for each logical message. The default prefix is **.

TERMINAL
specifies that the data is to be sent to the terminal that originated the transaction.

WAIT
specifies that control should not be returned to the application program until the output operation has been completed.

If WAIT is not specified, control returns to the application program when the output operation has started. A subsequent input or output command (terminal control, BMS, or batch data interchange) causes the application program to wait until the previous command has been completed.

Conditions

57 IGREQCD
occurs when an attempt is made to execute a SEND TEXT command after a SIGNAL data-flow control command with a request change direction (RCD) code has been received from an LUTYPE4 logical unit.

Default action: terminate the task abnormally.

39 IGREQID
occurs if the prefix specified in the REQID option on a BMS SEND command is different from that established by a previous REQID option, or by default for this logical message—REQID (**).

Default action: terminate the task abnormally.

65 INVPARTN
occurs if the specified partition is not defined in the partition set associated with the application program.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

200 Command not allowed for a distributed program link server program.

also occurs (RESP2 not set) in any of the following situations:

- Text data is output to the same partition or LDC as mapped data while a BMS logical message is active. If neither partitions nor LDCs are in use, text data is output to the same logical message as mapped data.
- During overflow processing, data is sent to a different LDC from the LDC that caused page overflow.
- The length of a header on a SEND TEXT command is negative.
- The length of a trailer on a SEND TEXT command is negative.

Default action: terminate the task abnormally.

35 TSIOERR
occurs if there is an irrecoverable temporary storage input/output error.

Default action: terminate the task abnormally.

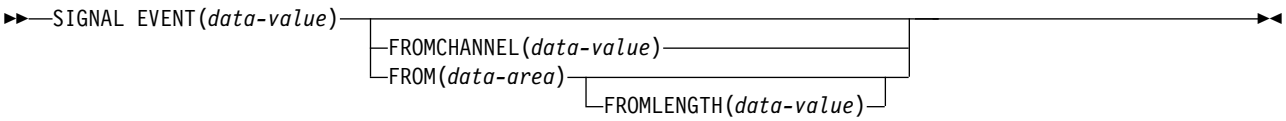
03 WRBRK
occurs if a SEND command is interrupted by the terminal operator pressing the ATTN key. It applies only to the 2741 Communication Terminal, and only if write break is supported for CICS.

Default action: ignore the condition.

SIGNAL EVENT

Identify a place in an application program where one or more events could be emitted.

SIGNAL EVENT



Conditions: CHANNELERR, EVENTERR, LENGERR

This command is threadsafe.

Description

The SIGNAL EVENT identifies a place in an application program where one or more events could be emitted. Events are emitted when the following conditions are satisfied:

- Event processing is active.
- There is least one matching capture specification enabled. A capture specification matches if it has a capture point of **SIGNAL EVENT**, and all its predicates evaluate to true.

SIGNAL EVENT has a primary predicate of EVENT, and allows secondary predicates on the **FROM** data-area or the **FROMCHANNEL** and its containers. The data in

any CICS Event emitted as a result of **SIGNAL EVENT** is defined in the Business Event that contains the matching capture specification.

Options

EVENT(data-value)
Specifies an identifier (1 – 32 characters) that identifies this **SIGNAL EVENT**.

The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is fewer than 32 characters, it is padded with trailing blanks up to 32 characters.

Event identifiers are always in EBCDIC. The allowable set of characters for event identifiers, listed above, includes some characters that do not have the same representation in all EBCDIC code pages. We therefore recommend that the characters used in naming them should be restricted to A-Z a-z 0-9 & : = , ; < > . - and _.

FROMCHANNEL(data-value)
Specifies the name (1 – 16 characters) of a channel containing data for this event. You can specify the channel name `DFHTRANSACTION` to use the transaction channel.

FROM(data-area)
Specifies a data area containing data for this event.

FROMLENGTH(data-value)
Specifies a fullword binary value that is the length in bytes of the **FROM** data area.

Conditions

122 CHANNELERR
RESP2 values:

2 The channel specified on the FROMCHANNEL option could not be found.

111 EVENTERR
RESP2 values:

6 The identifier specified on the EVENT option contains an incorrect character or combination of characters.

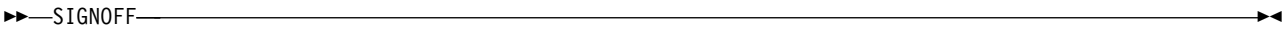
22 LENGERR
RESP2 values:

3 The length that you have specified in FROMLENGTH is not greater than zero.

SIGNOFF

Sign off from a terminal.

SIGNOFF



Condition: INVREQ

This command is threadsafe.

Description

SIGNOFF enables you to sign off from the terminal or principal facility that you previously signed on to. When sign-off is complete, the terminal reverts to the security capabilities and operator characteristics associated with the default user for this CICS region. The national language reverts to the national language of the default user, if defined, or the national language associated with the definition of the terminal.

When this command is executed, CICS immediately recognizes the sign-off and establishes the default attributes for the terminal. The transaction (and any associated task-related user exits, function shipping, or distributed transaction processing) may have invoked other resource managers (RMs), for example, IMS, DB2®, or VSAM. **It is unpredictable whether these other RMs recognize the sign-off before the transaction terminates.**

The default attributes apply for all RMs invoked by subsequent transactions at the terminal.

Conditions

16 INVREQ

RESP2 values:

1 No user is currently signed on. This could be because the CICS ESM is not initialized.

2 There is no terminal with this task.

3 This task's terminal has preset security.

4 Sign-off is attempted using transaction routing without using the CRTE transaction.

18 The CICS ESM interface is not initialized.

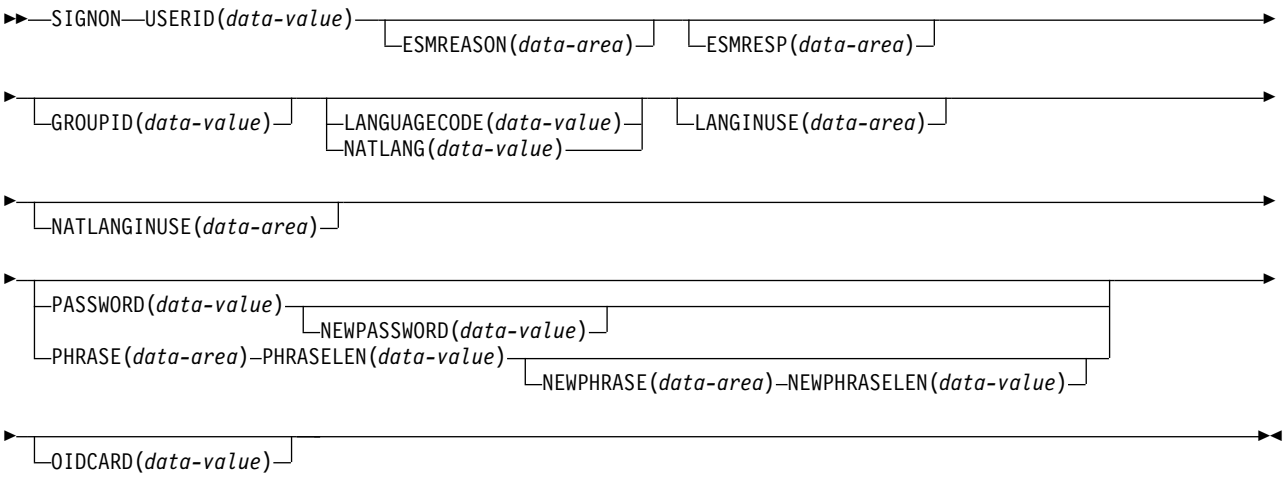
200 Command not allowed for a distributed program link server program.

Default action: terminate the task abnormally.

SIGNON

Sign on to a terminal.

SIGNON



Conditions: INVREQ, LENGERR, NOTAUTH, USERIDERR

This command is threadsafe.

Description

The SIGNON command enables your application to associate a new user ID with the current terminal. When you use the SIGNON command, the following rules apply:

- The sign-on operation is terminal related only. Sign-on has no meaning if the transaction does not have a terminal as its principal facility.
- When you issue an EXEC CICS SIGNON command, CICS modifies the state of the terminal that is the principal facility of the transaction that issues the command.
- Signon does not affect the user ID and security capabilities currently in effect for the transaction issuing the command. This is because:
  - A transaction's user ID and security capabilities are established at transaction-attach time. It is not possible to modify these subsequently during the life of the transaction.
  - All actions performed by a transaction (whether to a local or remote resource, or to a connected system) take place in the security context established at the time the transaction was attached.
- If authorization is required, you can sign on with either a valid password or a valid password phrase. However you cannot set a new password phrase using a password for authentication, nor can you set a new password using a password phrase for authentication.

If you supply an incorrect password on the **SIGNON** request, the invalid attempt count is increased for the user ID. If you supply multiple incorrect passwords on successive **SIGNON** requests, the user ID may be revoked by the external security manager. CICS issues message DFHXS1201 when you supply an incorrect

password on a **SIGNON** request. When you supply a correct password following one or more invalid attempts, CICS issues message DFHXS1206, which includes a count of the invalid attempts.

There is no implied sign-off with the SIGNON command. If your application program attempts to associate a new user with a terminal that already has a signed-on user ID, CICS returns an INVREQ (Resp2=9) error response. Note that there is no default value for the USERID option.

PASSWORD is used as a parameter which means that if CICS has a memory dump, the password may be visible. You should therefore clear the PASSWORD field as soon as possible after using it in a SIGNON command.

For more information on how CICS uses the USERID and GROUPID, see Verifying CICS users.

Options

If an optional input field contains all blanks, it is ignored.

ESMREASON(*data-area***)**
returns the reason code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF reason code.

ESMRESP(*data-area***)**
returns the response code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF return code.

GROUPID(*data-value***)**
assigns, to a RACF user group, the user that is being signed on. This overrides, for this session only, the default group name specified for the user in the RACF database.

LANGUAGECODE(*data-value***)**
specifies the national language that the user being signed on wants CICS to use. You specify the language as a standard 3-character IBM code. This is an alternative to the 1-character code that you specify on the NATLANG option.

See National language codes for possible values of the code.

LANGINUSE(*data-area***)**
the LANGINUSE option allows an application program to receive the national language chosen by the sign-on process. The language is identified as a standard three-character IBM code, instead of the one-character code used by NATLANGINUSE. It is an alternative to the existing NATLANGINUSE option.

See National language codes for possible values of the code.

NATLANG(*data-value***)**
specifies a 1-character field identifying the national language the user wants to use during the signed-on session.

See National language codes for possible values of the code.

NATLANGINUSE(*data-area***)**
specifies a one character the national language used during the signed-on session. The current implementation always returns the character E (U.S.

English), which corresponds to the language supplied in the NATLANG option. NATLANGINUSE corresponds to the following (in order of decreasing priority):

- The language supplied in the NATLANG option of the SIGNON command.
- The language associated with the user. This is specified in the ESM language segment.
- The language associated with the definition of the terminal.
- The language associated with the default USERID for the CICS region.
- The default language specified in the system initialization parameters.

See National language codes for possible values of the code.

NEWPASSWORD(*data-value***)**
specifies an optional 8-byte field defining a new password. This option is valid only if PASSWORD is also specified. You cannot enter a password phrase in this field.

NEWPHRASE(*data-area***)**
specifies an optional 1-to 8-character new password or a 9- to 100-character new password phrase required by the ESM. This option is valid only if PHRASE is also specified.

NEWPHRASELEN(*data-value***)**
specifies the length of the new password phrase as a fullword binary value. This option is valid only if NEWPHRASE is also specified.

OIDCARD(*data-value***)**
specifies an optional 65-byte field containing further security data from a magnetic strip reader (MSR) on 32xx devices.

PASSWORD(*data-value***)**
specifies an 8-byte password required by the external security manager (ESM).

PHRASE(*data-area***)**
specifies a optional 1-to 8-character password or a 9- to 100-character password phrase required by the ESM.

PHRASELEN(*data-value***)**
specifies the length of the password phrase as a fullword binary value. This option is valid only if PHRASE is also specified.

USERID(*data-value***)**
specifies the 8-byte sign-on USERID.

Conditions

16 INVREQ
RESP2 values:

2 A password cannot be used to change a password phrase or a password phrase cannot be used to change a password.

9 The terminal is already signed on.

10 No terminal is associated with this task.

11 This task's terminal has preset security.

12 The response from CICS security modules is unrecognized.

13 There is an unknown return code in ESMRESP from the external security manager; or the external security manager (ESM) is not active, or has failed in an unexpected way.

14 The required national language is not available.

15 Signon was attempted using transaction routing without using the CRTE transaction.

18 The CICS ESM interface is not initialized (SEC=NO specified as a System initialization parameter).

25 The terminal is of an invalid type.

26 An error occurred during SNSCOPE checking. The limit of MVS ENQ requests was reached.

27 The external security manager (ESM) is not active.

28 The required national language is invalid.

29 The user is already signed on. This relates to the sign-on scope checking.

200 Command not allowed for a distributed program link server program.

Default action: terminate the task abnormally.

22 LENGERR

RESP2 values:

1 PHRASELEN was out-of-range .

2 NEWPHRASELEN was out-of-range .

70 NOTAUTH

RESP2 values:

1 A password or password phrase is required.

2 The supplied password or password phrase is wrong.

3 A new password or password phrase is required.

4 The new password or password phrase is not acceptable.

5 An OIDCARD is required.

6 The supplied OIDCARD is wrong.

16 The USERID is not authorized to use this terminal.

17 The USERID is not authorized to use the application.

19 The USERID is revoked.

20 The USERID's access to the specified group has been revoked.

21 The sign-on failed during SECLABEL checking.

22 The sign-on failed because the ESM is not currently accepting sign-on.

23 The GROUPID is not known to the ESM.

24 The USERID is not contained in the GROUPID.

Default action: terminate the task abnormally.

69 USERIDERR

RESP2 values:

8 The USERID is not known to the external security manager.

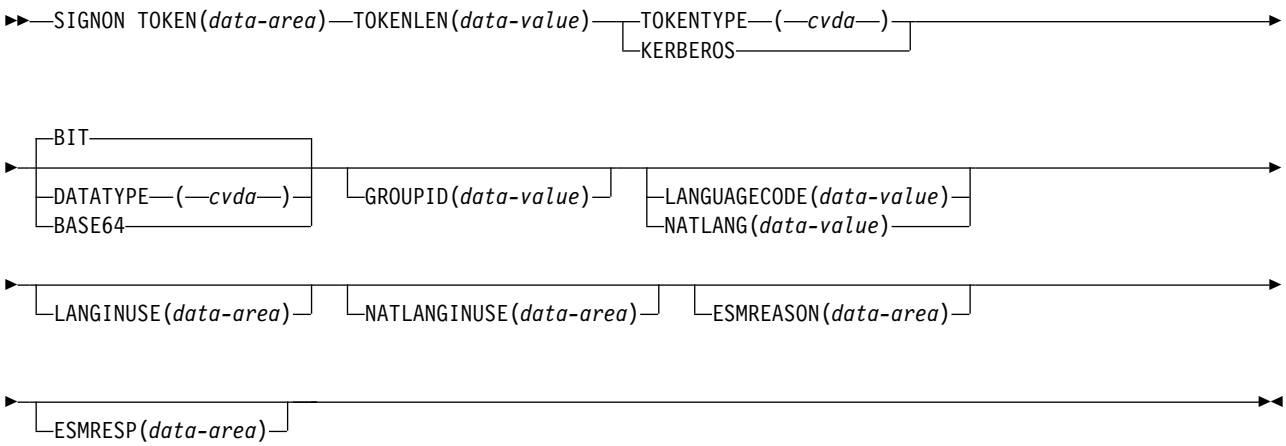
30 The USERID is all blanks or nulls.

Default action: terminate the task abnormally.

SIGNON TOKEN

Validate a security token and sign on to a terminal with the user ID extracted from the token.

SIGNON TOKEN



Conditions: INVREQ, LENGERR, NOTAUTH

This command is threadsafe.

Description

The SIGNON TOKEN command enables your application to validate a Kerberos security token, as determined by an external security manager (ESM), and associate a new user ID with the current terminal. When you use the SIGNON TOKEN command, the following applies:

- The sign-on operation is terminal related only. Sign-on has no meaning if the transaction does not have a terminal as its principal facility.
- When you issue an EXEC CICS SIGNON TOKEN command, CICS modifies the state of the terminal that is the principal facility of the transaction that issues the command.
- Sign-on does not affect the user ID and security capabilities currently in effect for the transaction issuing the command. This is because:
  - A transaction's user ID and security capabilities are established at transaction-attach time. It is not possible to modify these subsequently during the life of the transaction.
  - All actions performed by a transaction (whether to a local or remote resource, or to a connected system) take place in the security context established at the time the transaction was attached.
- The SIGNON TOKEN command uses the z/OS Security Server to verify that the token is a valid Kerberos token and that it can be used by the CICS region.
- If the ESM is RACF, the CICS region in which the command is run must be authorized by RACF so that the Kerberos principal for the token can be obtained. For more information, see Configuring RACF for Kerberos.

- There is no implied sign-off with the SIGNON TOKEN command. If your application program attempts to associate a new user with a terminal that already has a signed-on user ID, CICS returns an INVREQ (Resp2=9) error response.

For more information about how CICS uses the USERID and GROUPID, see Verifying CICS users.

For more information about a security failure of this command, see the error messages that are written to destination CSCS.

Options

If an optional input field contains all blanks, it is ignored.

DATATYPE
Specifies the type of data in the token. CVDA values are as follows:

BIT Bit data. This is the default value.

BASE64

Base64 encoded character data. The acceptable characters are A-Z a-z 0-9 + / =

If your character data is not in a US EBCDIC compatible character CCSID you must convert it. You can use the CONTAINER API to do the conversion.

ESMREASON(*data-area***)**
Returns the reason code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF reason code.

The ESM does not always return response and reason codes to CICS. Make sure that you check the EIBRESP and EIBRESP2 values that are returned by this command in addition to checking the ESMRESP and ESMREASON values.

ESMRESP(*data-area***)**
Returns the response code, in a fullword binary field, that CICS receives from the external security manager.

If the ESM is RACF, this field is the RACF return code.

The ESM does not always return response and reason codes to CICS. Make sure that you check the EIBRESP and EIBRESP2 values that are returned by this command in addition to checking the ESMRESP and ESMREASON values.

GROUPID(*data-value***)**
Assigns, to a RACF user group, the user that is being signed on. This overrides, for this session only, the default group name specified for the user in the RACF database.

LANGUAGECODE(*data-value***)**
Specifies the national language that the user being signed on wants CICS to use. You specify the language as a standard 3-character IBM code. This is an alternative to the 1-character code that you specify on the NATLANG option.

See Chapter 6, “National language codes,” on page 839 for possible values of the code.

LANGINUSE(*data-area***)**
The LANGINUSE option allows an application program to receive the national language chosen by the sign-on process. The language is identified as a standard three-character IBM code, instead of the one-character code used by NATLANGINUSE. It is an alternative to the existing NATLANGINUSE option.

See Chapter 6, “National language codes,” on page 839 for possible values of the code.

NATLANG(*data-value***)**
Specifies a 1-character field identifying the national language the user wants to use during the signed-on session.

See Chapter 6, “National language codes,” on page 839 for possible values of the code.

NATLANGINUSE(*data-area***)**
Specifies a one-character code for the national language used during the signed-on session. The current implementation always returns the character E (U.S. English), which corresponds to the language supplied in the NATLANG option. NATLANGINUSE corresponds to the following (in order of decreasing priority):

- The language supplied in the NATLANG option of the SIGNON TOKEN command.
- The language associated with the user. This is specified in the ESM language segment.
- The language associated with the definition of the terminal.
- The language associated with the default USERID for the CICS region.
- The default language specified in the system initialization parameters.

See Chapter 6, “National language codes,” on page 839 for possible values of the code.

TOKEN(*data-area***)**
A token that has been obtained from a Security Token Service (STS).

TOKENLEN(*data-value***)**
The length of the token as a fullword binary value.

TOKENTYPE(*cvda***)**
Indicates the type of token.

KERBEROS
The token is a Kerberos token.

Conditions

16 INVREQ
RESP2 values:

9 The terminal is already signed on.

10 No terminal is associated with this task.

11 This task's terminal has preset security.

12 The response from CICS security modules is unrecognized.

13 There is an unknown return code in ESMRESP from the external security manager (ESM), or the ESM is not active, or has failed in an unexpected way.

14 The required national language is not available.

15 Signon was attempted using transaction routing without using the CRTE transaction.

18 The CICS ESM interface is not initialized (SEC=NO specified as a System initialization parameter).

25 The terminal is of an invalid type.

26 An error occurred during SNSCOPE checking. The limit of MVS ENQ requests was reached.

27 The ESM is not active.

28 The required national language is invalid.

29 The user is already signed on. This relates to the sign-on scope checking.

30 The ESM is not responding.

31 A CVDA value other than KERBEROS was specified for **TOKENTYPE**.

32 A CVDA value other than BASE64 or BIT was specified for **DATATYPE**.

36 A data-type of BASE64 was specified, but **TOKEN** does not contain BASE64 data.

40 The key distribution center is not started or is terminating.

41 The key distribution center is not responding.

47

The ESM does not have a user ID defined for the Kerberos principal that is associated with the token.

50

The data specified in **TOKEN** is not a Kerberos token

200 Command not allowed for a distributed program link server program.

Default action: terminate the task abnormally.

22 LENGERR

RESP2 values:

45 The length of the Kerberos token exceeds the maximum value of 65535.

70 NOTAUTH

RESP2 values:

16 The USERID is not authorized to use this terminal.

17 The USERID is not authorized to use the application.

19 The USERID is revoked.

20 The USERID's access to the specified group has been revoked.

21 The sign-on failed during SECLABEL checking.

22 The sign-on failed because the ESM is not currently accepting sign-on.

23 The GROUPID is not known to the ESM.

24 The USERID is not contained in the GROUPID.

40 The ESM does not authorize the request to verify the token. Error messages are written to destination CSCS.

42 A Kerberos request cannot be completed because the associated ticket has expired.

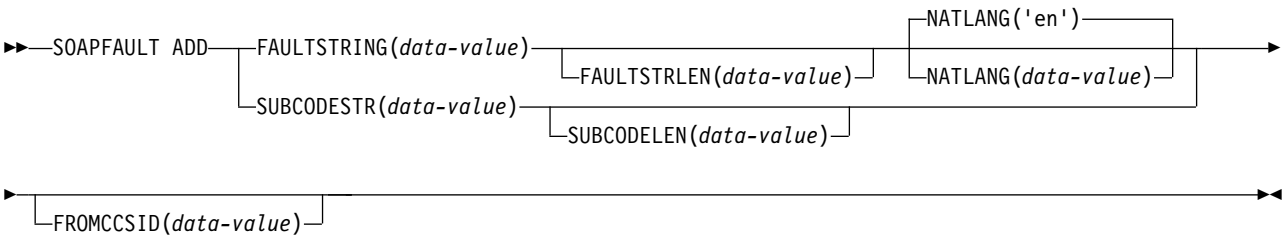
43 The authenticator has expired.

Default action: terminate the task abnormally.

SOAPFAULT ADD

Adds information to an existing SOAPFAULT object. You can use this command only in a program that is invoked from a CICS-supplied SOAP message handler.

SOAPFAULT ADD



Conditions: CHANNELERR, INVREQ, LENGERR

This command is threadsafe.

Description

This command adds information to a SOAPFAULT object that was created with an earlier **SOAPFAULT CREATE** command. The information that can be added comprises:

- A subcode
- A fault string for a particular national language. If the fault already contains a fault string for the language, then this command replaces the fault string for that language. In SOAP 1.1, only the fault string for the original language is used.

This command requires information that is held in containers on the channel of the supplied SOAP message handler. To use this command, you must have access to the channel. Only the following types of programs have this access:

- Programs that are invoked as SOAP header handlers
- Programs that are invoked directly from a CICS-supplied SOAP message handler
- Programs deployed with the CICS web services assistant that have a channel interface. Programs with a COMMAREA interface do *not* have access to the channel.
- Programs that are using global user exits to monitor web services.

Many of the options on this command apply to SOAP 1.1 and SOAP 1.2 faults, although their behavior is slightly different for each level of SOAP. Other options apply to one SOAP level or the other, but not to both, and if you specify any of them when the message uses a different level of SOAP, the command raises an INVREQ condition. To help you determine which SOAP level applies to the message, container DFHWS-SOAPLEVEL contains a binary fullword with one of the following values:

1 The request or response is a SOAP 1.1 message.

2 The request or response is a SOAP 1.2 message.

10 The request or response is not a SOAP message.

Options

SUBCODESTR(*data-value*)
Specifies the contents of a <Subcode> element that is to be added to the SOAPFAULT object. The subcode can be up to 64 characters in length, and must be an XML qualified name (QName).
- For SOAP 1.1, this option is ignored.
- For SOAP 1.2, this option supplies the contents of the <Subcode> element.

SUBCODELEN(*data-value*)
specifies the length, as a fullword binary value, of the <Subcode> element specified in the SUBCODESTR option.

FAULTSTRING(*data-value*)
Specifies a human-readable explanation of the fault. The FAULTSTRING value can be up to 2056 characters in length.
- For SOAP 1.1, this option supplies the contents of the <faultstring> element.
- For SOAP 1.2, this option supplies the contents of the <Reason> element.

FAULTSTRLEN(*data-value*)
Specifies the length, as a fullword binary value, of the FAULTSTRING option.

FROMCCSID(*data-value*)
Specifies, as a fullword binary number, the current Coded Character Set Identifier (CCSID) of the character data to be put into the SOAP fault. If this option is not specified, CICS uses the value that is specified in the **LOCALCCSID** system initialization parameter. For more information about CCSIDs, and a list of the CCSIDs supported by CICS, see CICS-supported conversions.

NATLANG(*data-value*)
Specifies an 8 character field containing the national language used for the FAULTSTRING. The language is specified using the XML 1.0 language identification. The default value is 'en' (English).

When the language identifier is shorter than eight characters, you must pad it on the right with space characters in the character set specified in the FROMCCSID option (or the CICS LOCALCCSID). For example, if you specify the UTF-8 character set with FROMCCSID(1208), you must pad the NATLANG value with X'20' characters.

This option is used only for SOAP 1.2 faults.

Conditions

122 CHANNELERR
RESP2 values are:

3 The channel where this SOAPFAULT object is being added is read-only.

16 INVREQ
RESP2 values are:

3 The command was issued outside the environment of a CICS-supplied SOAP handler.

7 No SOAP fault present

11 Invalid subcode

22 LENGERR
RESP2 values are:

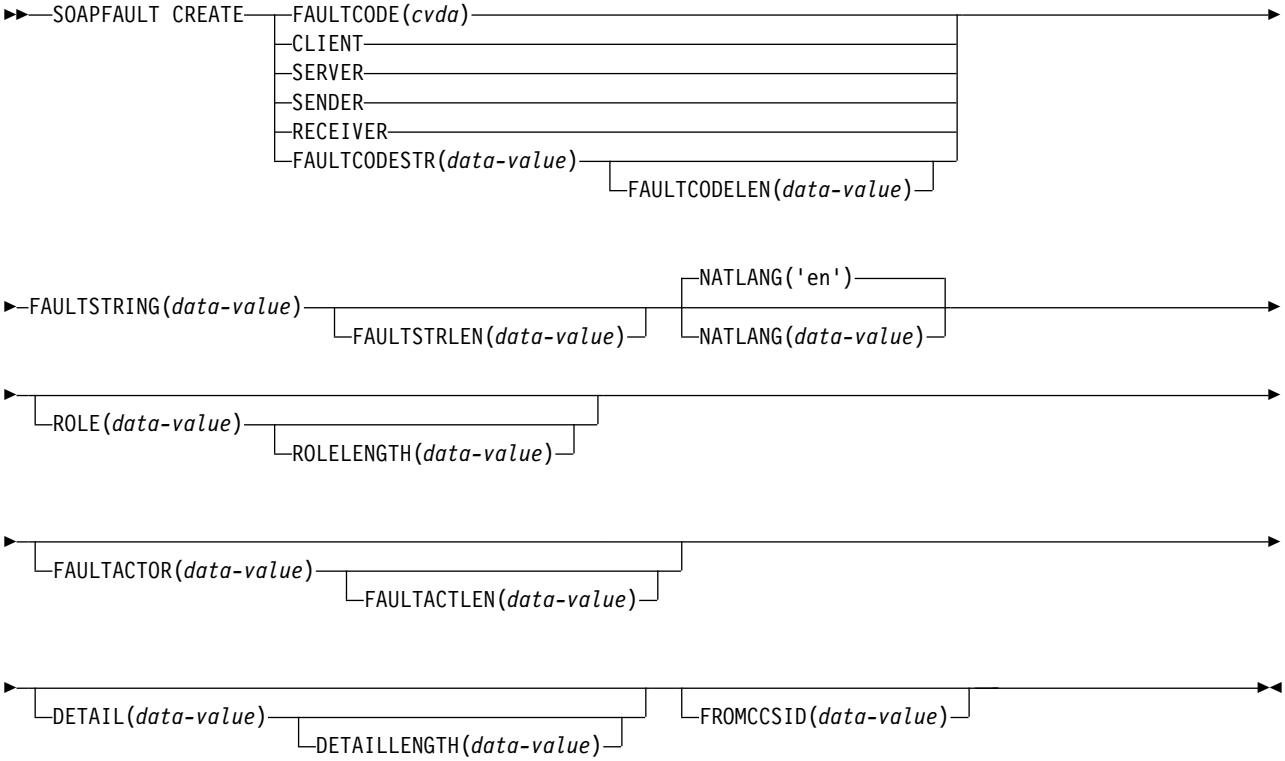
6 The FAULTSTRLEN value is invalid

10 The SUBCODELEN value is invalid

SOAPFAULT CREATE

Create a SOAP fault response message that is returned as the response to a web service request. The web service application response is not processed.

SOAPFAULT CREATE



Conditions: CCSIDERR, CHANNELERR, INVREQ, LENGERR

This command is threadsafe.

Description

SOAPFAULT CREATE creates a SOAP fault. If a SOAP fault already exists in the context of the SOAP message that is being processed by the message handler, the existing fault is overwritten. This command can be used only in programs attached from a supplied SOAP handler. Axis2 applications hosted in a Java pipeline in CICS cannot make use of this command.

This command requires information that is held in containers on the channel of the supplied SOAP message handler. To use this command, you must have access to the channel. Only the following types of programs have this access:

- Programs that are invoked as SOAP header handlers
- Programs that are invoked directly from a CICS-supplied SOAP message handler
- Programs deployed with the CICS web services assistant that have a channel interface. Programs with a COMMAREA interface do *not* have access to the channel.
- Programs that are using global user exits to monitor web services.

Many of the options on this command apply to SOAP 1.1 and SOAP 1.2 faults, although their behavior is slightly different for each level of SOAP. Other options apply to one SOAP level or the other, but not to both, and if you specify any of them when the message uses a different level of SOAP, the command raises an INVREQ condition. To help you determine which SOAP level applies to the message, container DFHWS-SOAPLEVEL contains a binary fullword with one of the following values:

1 The request or response is a SOAP 1.1 message.

2 The request or response is a SOAP 1.2 message.

10 The request or response is not a SOAP message.

Options

DETAIL(*data-value*)
Specifies the following contents:
- For SOAP 1.1, this option supplies the contents of the `<detail>` element of the SOAP fault.
- For SOAP 1.2, this option supplies the contents of the `<Detail>` element of the SOAP fault.

It should contain either one or more valid namespace-qualified XML elements, or white space. Refer to the appropriate SOAP specifications for a full description of the valid content of the element.

The element carries application-specific error information related to the `<Body>` element, and is used when the contents of the `<Body>` element could not be successfully processed. For SOAP 1.1, the `<detail>` element must be present if the contents of the `<Body>` element could not be successfully processed; for SOAP 1.2, the `<Detail>` element is optional.

If the SOAPFAULT CREATE command is issued in a header handler program, the detail content is carried in a header block for SOAP 1.1. For SOAP 1.2 the `<Detail>` element is carried within the SOAP fault.

DETAILLENGTH(*data-value*)
Specifies the length, as a fullword binary value, of the `DETAIL` option.

FAULTACTLEN(*data-value*)
Specifies the length, as a fullword binary value, of the `FAULTACTOR` option.

FAULTACTOR(*data-value*)
Specifies the following contents:
- For SOAP 1.1, this option supplies the contents of the `<faultactor>` element.
- For SOAP 1.2, this option supplies the contents of the `<Node>` element.

The `FAULTACTOR` option can be up to 2056 characters in length, and must be a valid URI (`anyURI`).

FAULTCODE(*cvda*)
CVDA values are as follows:

CLIENT
For SOAP 1.1, the CVDA value of `Client` should be specified. However, if you specify `Sender` for a SOAP 1.1 fault response message, CICS will use the CVDA value of `Client`.

SENDER
For SOAP 1.2, the CVDA value of `Sender` should be specified. However, if you specify `Client` for a SOAP 1.2 fault response message, CICS will use the CVDA value of `Sender`.

SERVER
For SOAP 1.1, the CVDA value of `Server` should be specified. However, if you specify `Receiver` for a SOAP 1.1 fault response message, CICS will use the CVDA value of `Server`.

RECEIVER
For SOAP 1.2, the CVDA value of `Receiver` should be specified. However, if you specify `Server` for a SOAP 1.2 fault response message, CICS will use the CVDA value of `Receiver`.

FAULTCODELEN(*data-value*)
Specifies the length, as a fullword binary value, of the `FAULTCODESTR` option.

FAULTCODESTR(*data-value*)
Specifies a user-defined SOAP fault code for a SOAP 1.1 message. The fault code can be up to 64 characters in length, and must be an XML qualified name (`QName`). The use of the dot (.) character to separate fault code values is not supported. For SOAP 1.1, this option supplies the contents of the `<faultcode>` element.

If you set this option for a SOAP 1.2 message, an INVREQ condition occurs.

FAULTSTRING(*data-value*)
Specifies a human-readable explanation of the fault. The FAULTSTRING value can be up to 2056 characters in length.
- For SOAP 1.1, this option supplies the contents of the `<faultstring>` element.
- For SOAP 1.2, this option supplies the contents of the `<Reason>` element.

FAULTSTRLEN(*data-value*)
Specifies the length, as a fullword binary value, of the `FAULTSTRING` option.

FROMCCSID(*data-value*)
Specifies, as a fullword binary number, the current Coded Character Set Identifier (CCSID) of the character data to be put into the SOAP fault. If this option is not specified, CICS uses the value that is specified in the **LOCALCCSID** system initialization parameter. For more information about CCSIDs, and a list of the CCSIDs supported by CICS, see CICS-supported conversions.

NATLANG(*data-value*)
Specifies an 8 character field containing the national language used for the `FAULTSTRING`. The language is specified using the XML 1.0 language identification. The default value is `'en'` (English).

When the language identifier is shorter than eight characters, you must pad it on the right with space characters in the character set specified in the FROMCCSID option (or the CICS LOCALCCSID). For example, if you specify the UTF-8 character set with FROMCCSID(1208), you must pad the NATLANG value with `X'20'` characters.

This option is used only for SOAP 1.2 faults.

ROLE(*data-value*)
Specifies the URI that describes the role of the SOAP node that generated the fault. The ROLE option can be up to 2056 characters in length, and must be a valid URI (XML type anyURI).
- For SOAP 1.1, this option is ignored.
- For SOAP 1.2, this option supplies the contents of the <Role> element.

ROLELENGTH(*data-value*)
Specifies the length, as a fullword binary value, of the ROLE option.

Conditions

123 CCSIDERR
RESP2 values are:

13 An invalid CCSID has been specified.

14 An unsupported CCSID has been specified.

122 CHANNELERR
RESP2 values are:

3 The channel where this SOAPFAULT object is being created is read-only.

16 INVREQ
RESP2 values are:

3 The command was issued outside the environment of a CICS-supplied SOAP handler.

11 The FAULTCODE specified is invalid or FAULTCODESTR was specified for a SOAP 1.2. fault.

13 The DETAIL option does not contain valid namespace-qualified XML or whitespace.

22 LENGERR
RESP2 values are:

5 The FAULTCODELEN value is invalid

6 The FAULTSTRLEN value is invalid

7 The ROLELENGTH value is invalid

8 The FAULTACTLEN value is invalid

9 The DETAILLENGTH value is invalid

SOAPFAULT DELETE

Deletes an existing SOAPFAULT object. You can use it only in a program that is invoked from a CICS-supplied SOAP message handler.

SOAPFAULT DELETE



Conditions: CHANNELERR, INVREQ, NOTFND

This command is threadsafe.

Description

This command deletes a SOAPFAULT object that was created with an earlier **SOAPFAULT CREATE** command.

This command requires information that is held in containers on the channel of the supplied SOAP message handler. To use this command, you must have access to the channel. Only the following types of programs have this access:

- Programs that are invoked as SOAP header handlers
- Programs that are invoked directly from a CICS-supplied SOAP message handler
- Programs deployed with the CICS web services assistant that have a channel interface. Programs with a COMMAREA interface do *not* have access to the channel.
- Programs that are using global user exits to monitor web services.

Conditions

122 CHANNELERR

3 The channel where this SOAPFAULT object is being deleted is read-only.

16 INVREQ

RESP2 values are:

3 The function was called when a CICS-supplied SOAP node was not in use.

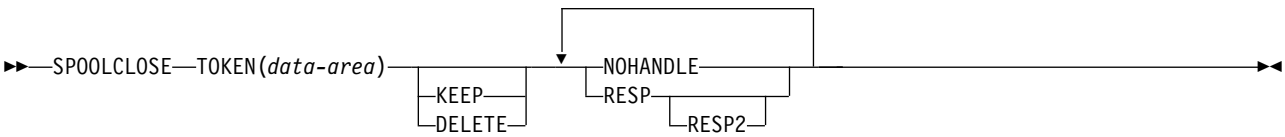
13 NOTFND

2 There is no SOAPFAULT object to delete.

SPOOLCLOSE

Close a spool report.

SPOOLCLOSE TOKEN



Conditions: ALLOCERR, INVREQ, NOSPOOL, NOSTG, NOTFND, NOTOPEN, STRELERR

Description

The SPOOLCLOSE command closes a CICS spool report and, optionally, changes its retention characteristics. If more than one transaction is trying to read reports from JES, SPOOLCLOSE should **not** be immediately followed by SPOOLOPEN. It should be followed by a WAIT, so that other transactions can use the interface.

A default disposition is taken if both KEEP and DELETE are omitted from the SPOOLCLOSE command, or if the report is closed implicitly by a SYNCPOINT or RETURN command:

- When an INPUT report is explicitly closed by a SPOOLCLOSE command, the default disposition is DELETE.
- In all other cases, the default disposition is KEEP.

Options

DELETE
For an INPUT report, DELETE specifies that the **next** report is to be read on the subsequent OPEN INPUT.

For an OUTPUT report, DELETE specifies that the report is to be purged.

Note: When a JCL job is submitted using the internal reader (INTRDR) with the DELETE option specified, the job is sometimes run before the output is deleted.

KEEP
For an INPUT report, KEEP specifies that the report is to be read again when SPOOLOPEN INPUT is next issued.

For an OUTPUT report, KEEP specifies that the report is to be sent to its destination node.

TOKEN(*data-area***)**
specifies the 8-character CICS-allocated token used to identify a report.

Conditions

Note: There are no default actions.

85 ALLOCERR
occurs in any of the following situations:

- Dynamic allocation has rejected a request to allocate an input data set.

  RESP2 gives the dynamic allocation response code that denotes this error. The first two characters are the information reason code (S99INFO), and the second two are the error reason code (S99ERROR), as defined in the *z/OS: MVS Programming: Authorized Assembler Services Guide, SA22-7608*.

16 INVREQ
RESP2 values:

4 Unsupported language.

8 Unsupported function.

40 Subsystem interface already enabled.

Note: Errors 1024 and over are internal, and should not occur. If one of these error codes is returned, contact your IBM support center.

80 NOSPOOL
RESP2 values:

4 No subsystem present.

8 Interface being disabled; CICS is quiescing.

12 Interface has been stopped.

42 NOSTG
occurs in any of the following situations:

- A GETMAIN has failed within the JES interface subtask (DFHPSPSS).
  RESP2 gives the GETMAIN register 15 return code.

13 NOTFND
RESP2 values:

1024 Input or output function has been corrupted, and SPOOLCLOSE could not complete.

19 NOTOPEN
RESP2 values:

8 Data set has not been opened.

86 STRELERR
occurs in any of the following situations:

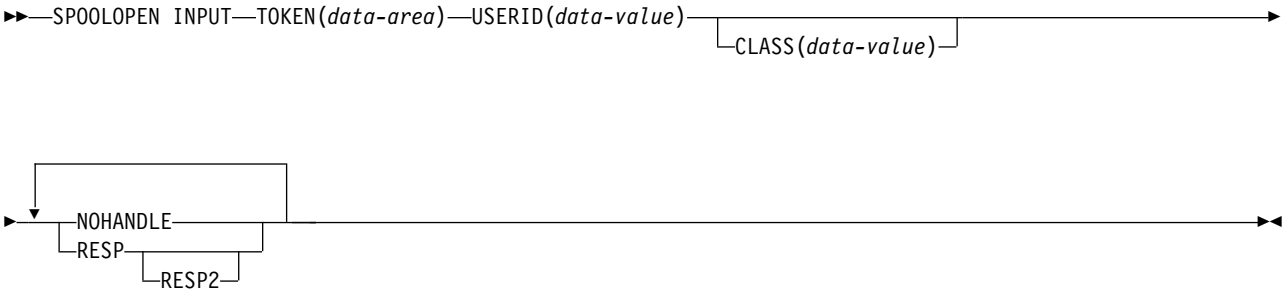
- A FREEMAIN has failed within the JES interface subtask (DFHPSPSS).
  RESP2 gives the FREEMAIN register 15 return code.

Restriction: You must specify the RESP or NOHANDLE option on the **EXEC CICS SPOOLCLOSE** command.

SPOOLOPEN INPUT

Open a spool report.

SPOOLOPEN INPUT



Conditions: ALLOCERR, ILLOGIC, INVREQ, NOSPOOL, NOSTG, NOTAUTH, NOTFND, NOTOPEN, OPENERR, SPOLBUSY, SPOLERR, STRELERR

Description

The SPOOLOPEN INPUT command opens a spool report for input from the system spooler to CICS.

It prepares to get (read) an existing spool data set directly using external writer name (USERID) and specified class.

Another task could have allocated a spool file for input. In this case, you should retry after a suitable time interval.

When this command has been successfully executed, you should read the report and proceed to CLOSE as soon as possible, in order to permit other users to use the JES single thread. If SPOOLCLOSE is not issued before transaction end or SYNCPOINT, CICS performs an implicit SPOOLCLOSE KEEP, and writes a message to CSMT to alert the system programmer to the possible unnecessary retention of resources. You should not SPOOLOPEN a data set using this command until you are prepared to process it completely.

This command, if successful, returns a token, which is used later to identify the report in SPOOLREAD and SPOOLCLOSE commands.

Options

CLASS(*data-value***)**
specifies a 1-character class designation. The CLASS operand can be used as a selection parameter for input reports. If it is omitted, the first report for the specified USERID is obtained, regardless of its class.

TOKEN(*data-area***)**
specifies the 8-character CICS-allocated token used to identify a report.

USERID(*data-value***)**
specifies the 8-character user identifier. It must begin with the same 4 characters as the CICS generic applid, so that CICS can check that users are not attempting to access data sets not intended for their CICS system.

Conditions

Note: There are no default actions.

85 ALLOCERR
occurs in any of the following situations:

- Dynamic allocation has rejected a request to allocate an input data set.

  RESP2 gives the dynamic allocation response code that denotes this error. The first two characters are the information reason code (S99INFO), and the second two are the error reason code (S99ERROR), as defined in the *z/OS: MVS Programming: Authorized Assembler Services Guide, SA22-7608*.

21 ILLOGIC
RESP2 values:

3 Invalid CLASS value specified.

16 INVREQ
RESP2 values:

4 Unsupported language.

8 Unsupported function.

16 USERID missing.

36 INPUT|OUTPUT missing.

40 Subsystem interface already enabled.

Note: Errors 1024 and over are internal, and should not occur. If one of these error codes is returned, contact your IBM Support Center.

80 NOSPOOL
RESP2 values:

4 No subsystem present.

8 Interface being disabled; CICS is quiescing.

12 Interface has been stopped.

42 NOSTG
occurs in any of the following situations:

- A GETMAIN has failed within the JES interface subtask (DFHPSPSS).

  RESP2 gives the GETMAIN register 15 return code.

70 NOTAUTH
occurs in any of the following situations:

- An application has issued a SPOOLOPEN INPUT command with an unauthorized USERID. For the USERID to be authorized, its first four characters must match the first four characters of the current CICS applid id.

13 NOTFND
RESP2 values:

4 No data sets could be located for retrieval for the specified external writer name; or the data set exists, but it is in HELD status.

Can also be returned if the CICS region USERID does not have ALTER access to the appropriate PROFILE in the JESSPOOL class.

1024 Input or output function has been corrupted, and SPOOLCLOSE could not complete.

19 NOTOPEN
RESP2 values:

8 Data set has not been opened or a task which has not issued the SPOOLOPEN for a spool data set has attempted to access it.

1024 Subtask OPEN macro failure.

87 OPENERR
RESP2 values:

4 A VSAM SHOWCB macro failed to return the lengths of the VSAM control blocks used to access the JES spool file.

Also occurs (RESP2 not set) in any of the following situations:

- An internal error occurred during SPOOLOPEN processing that has forced the request to fail.

88 SPOLBUSY
RESP2 values:

4 Interface already in use by another task.

8 Interface already in use by current task.

Also occurs (RESP2 not set) in any of the following situations:

- The JES/input single thread within the JES interface was not available.

89 SPOLERR
occurs in any of the following situations:

- The subsystem interface macro (IEFSSREQ) has failed. No input data set name was selected.

  RESP2 gives the 'IEFSSREQ' response code.

86 STRELERR
occurs in any of the following situations:

- A FREEMAIN has failed within the JES interface subtask (DFHPSPSS).

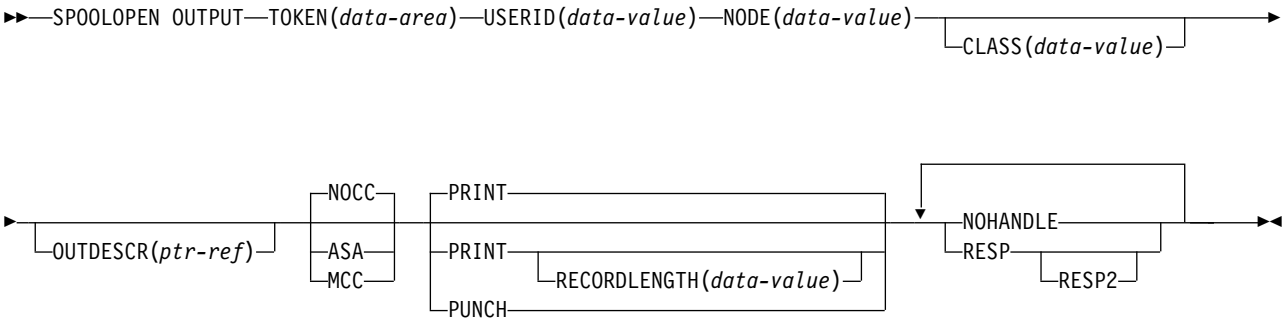
  RESP2 gives the FREEMAIN register 15 return code.

Restriction: You must specify the RESP or NOHANDLE option on the **EXEC CICS SPOOLOPEN** command.

SPOOLOPEN OUTPUT

Open a spool report for output.

SPOOLOPEN OUTPUT



Conditions: ALLOCERR, ILLOGIC, INVREQ, LENGERR, NODEIDERR, NOSPOOL, NOSTG, NOTFND, NOTOPEN, OPENERR, OUTDESCERR, SPOLBUSY, STRELERR

Description

The SPOOLOPEN OUTPUT command opens a spool report for output from CICS to the system spooler and defines its characteristics.

It results in a dynamic allocation of the output file using the nodeid to specify the remote destination and the userid to specify the remote user. As this is a multithread output request, requesters of this service could interleave. This SPOOLOPEN OUTPUT command enables users to acquire the token for a report that it expects to create (write). This token is used to identify the report in later SPOOLWRITE and SPOOLCLOSE commands.

When printing on a local device, use the NOCC | ASA | MCC options to control output formatting. If you do not specify a format, the default value of NOCC is used. NODE and USERID can be used to write the data set directly to the local spool file only if specified with a value of '*'.

If you do not issue SPOOLCLOSE before the end of the transaction, CICS performs an implicit SPOOLCLOSE DELETE and writes a message to CSMT to alert you to the possible unnecessary retention of resources.

Note: If you retrieve a formatted data set, the system spooler could have changed the data set format. For example, the system spooler could have converted an MCC format data set to ASA format during data set creation. This does not affect the final printed output.

Options

ASA
specifies that the report has each record prefixed with an ASA carriage-control character, and this character must be used by the operating system to control formatting when the report is printed.

CLASS(*data-value***)**
specifies a 1-character class designation. If it is omitted, class A is assumed.

MCC
specifies that the report has each record prefixed with an IBM machine command code carriage-control character, and this character must be used by the operating system to control formatting when the report is printed.

NOCC
specifies that the report has no internal formatting controls. When the report is printed, the operating system prefixes each record with a carriage-control character that causes page skipping according to the default operating system lines-per-page value.

NODE(*data-value***)**
specifies the 8-character identifier of a destination node that the system spooler uses to route the file. It is a sender field. If you want to specify the local spool file and to enable the OUTDESCR operand to override the NODE and USERID operands, code NODE('*') and also USERID('*'). (Do not use NODE('*') with any other userid.) Otherwise, code the actual NODE, which is the name of the operating system (for example, MVS, VM) as that system is known to VTAM in the MVS system in which your CICS is executing. NODE(LOCAL) is also a valid specification.

Validity checking is performed for NODE. Checks are made for blanks (X'40'), and nulls (X'00').

OUTDESCR(*ptr-ref***)**
(MVS/SP—JES2 Version 3, or JES3 Version 4.2.1 only, or a later upward-compatible release) specifies a pointer variable to be set to the address of a field that contains the address of a string of parameters to the OUTPUT statement of JCL. This is called double indirect addressing. The user must set up the pointer, the address field, and the string. This means that the OUTDESCR option cannot be used from within CECI. The format of the string is:

```
Offset Length Contents
0 4 Length (n) of following text string
4 n OUTPUT statement parameters
```

The parameters use the same keywords and values as the OUTPUT statement but the syntax varies slightly. The following is the format of the OUTDESCR parameter string:

```
keyword1(value1) [keyword2(value2)]
[keyword3(value3,value4)] ...
```

This corresponds to the following OUTPUT statement parameter string:

```
keyword1=value1 [keyword2=value2]
[keyword3=(value3,value4)] ...
```

For details of valid keywords and values, see the *z/OS TSO/E System Programming Command Reference*.

The OUTDESCR operand:

- Can override the NODE and USERID operands only if they are specified with a value of '*'.
- Cannot override the CLASS operand, even if it is omitted and defaults to class A.

Use this operand to set additional attributes for the spool data set.

PRINT
allows large records (maximum 32 760 bytes) to be written to the spool. This is the default setting. This is included for compatibility with the spool support provided with CICS Transaction Server for z/OS.

PUNCH
must be specified if the CLASS parameter for the output data set implies punch, and the data set is destined for a VM/RSCS node. This ensures that the record length indicator is set to 80, which is a requirement of VM/RSCS for punch files.

RECORDLENGTH(*data-value***)**
specifies, as a halfword binary variable, the maximum length of record to write to a print data set. The default value is 32 760.

TOKEN(*data-area***)**
specifies the 8-character CICS-allocated token used to identify a report.

USERID(*data-value***)**
specifies the 8-character identifier of the destination userid that processes the report. The report carries this identifier, which is used to select the report at its destination. It is a sender field and must be declared with a length of 8 characters.

If you want to specify the local spool file and to enable the OUTDESCR operand to override the NODE and USERID operands, code USERID('*') and also NODE('*'). Otherwise, code the actual USERID. The meaning of USERID varies with the operating system. In VM, it is a particular user; in MVS, it might be a JES external writer or another JES destination, a TSO user, or another job executing on that system. One such destination is the JES internal reader, which normally has the reserved name INTRDR. If you code an actual USERID, do not use NODE('*'); code the actual NODE instead.

The USERID parameter is equivalent to the WRITER parameter in JES.

Validity checking is performed for USERID. Checks are made for blanks (X'40'), and nulls (X'00').

Sending the internal reader buffer directly to JES: Instead of waiting for the buffer in your address space to fill up, send the contents of the internal reader buffer directly to JES by coding as your last record:

```
/*EOF
```

This control statement delimits the job in the data set, and makes it eligible for immediate processing.

For more information about using the internal reader, and about other /* control statements, see the *z/OS MVS JCL User's Guide*.

Restriction: You must specify the RESP or NOHANDLE option on the **EXEC CICS SPOOLOPEN** command.

Conditions

Note: There are no default actions.

85 ALLOCERR
occurs in any of the following situations:

- Dynamic allocation has rejected a request to allocate an input data set.

  RESP2 gives the dynamic allocation response code that denotes this error. The first two characters are the information reason code (S99INFO), and the second two are the error reason code (S99ERROR), as defined in the *z/OS MVS Programming: Assembler Services Guide*.

21 ILLOGIC
occurs in any of the following situations:

- Invalid CLASS value specified.

16 INVREQ
RESP2 values:

4 Unsupported language.

8 Unsupported function.

16 USERID missing.

20 NODE missing.

36 INPUT|OUTPUT missing.

40 Subsystem interface already enabled.

Note: Errors 1024 and over are internal, and should not occur. If one of these error codes is returned, contact your IBM support center.

44 Error in the OUTDESCR string.

48 OUTDESCR specified but function not available (wrong level of CICS or JES).

52 OUTDESCR specified but bad pointer found on keyword or in OUTDESCR condition.

22 LENGERR
occurs in any of the following situations:

- RECORDLENGTH not in the range 0 through 32760. RESP2 shows the incorrect value.

90 NODEIDERR
occurs in any of the following situations:

- JES cannot identify the NODE/USERID combination specified on SPOOLOPEN OUTPUT.

  RESP2 gives the dynamic allocation response code that denotes this error. The first two characters are the information reason code (S99INFO), and the second two are the error reason code (S99ERROR), as defined in the *z/OS MVS Programming: Assembler Services Guide*.

80 NOSPOOL
RESP2 values:

4 No subsystem present.

8 Interface being disabled; CICS is quiescing.

12 Interface has been stopped.

42 NOSTG
occurs in any of the following situations:

- A GETMAIN has failed within the JES interface subtask(DFHPSPSS).

  RESP2 gives the GETMAIN register 15 return code.
- Insufficient storage was available to copy and modify the OUTDESCR argument string.

13 NOTFND
RESP2 values:

4 No data sets could be located for retrieval for the specified external writer name.

19 NOTOPEN
RESP2 values:

8 Data set has not been opened.

1024 Subtask OPEN macro failure.

87 OPENERR
RESP2 values:

4 A VSAM SHOWCB macro failed to return the lengths of the VSAM control blocks used to access the JES spool file.

Also occurs (RESP2 not set) in any of the following situations:

- An internal error occurred during SPOOLOPEN processing that has forced the request to fail.

96 OUTDESCRERR
occurs in any of the following situations:

- The macro OUTADD or OUTDEL (invoked as a result of the OUTDESCR specification) failed.

  RESP2 gives the reason code from the OUTADD or OUTDEL macro. See the *z/OS MVS Programming: Assembler Services Guide* for descriptions of codes.

88 SPOLBUSY
RESP2 values:

4 Interface already in use by another task.

8 Interface already in use by current task.

Also occurs in the following situation:

- The JES/input single thread within the JES interface was not available.

86 STRELERR
occurs in the following situation:

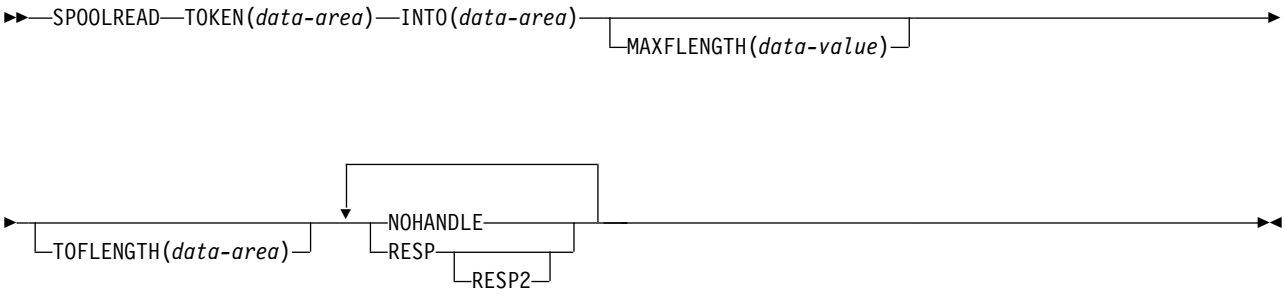
- A FREEMAIN has failed within the JES interface subtask (DFHPSPSS).

  RESP2 gives the FREEMAIN register 15 return code.

SPOOLREAD

The SPOOLREAD command obtains the next record from the system spooler.

SPOOLREAD



Conditions: ALLOCERR, ENDFILE, ILLOGIC, INVREQ, LENGERR, NOSPOOL, NOSTG, NOTFND, NOTOPEN, SPOLBUSY, SPOLERR, STRELERR

Options

INTO(*data-area***)**
specifies the data area for the variable-length data. It is a receiver field.

MAXFLENGTH(*data-value***)**
specifies, as a fullword binary variable, the maximum length of data transferred. This is set by the user on input. The limit of **length** is 32 760 bytes. This is the maximum size of the CICS buffer used to read a record.

TOFLENGTH(*data-area***)**
specifies, as a fullword binary variable, the length of the data that is transferred. This is set by CICS on input. It is optional and, if it is omitted, you are not notified of the actual length of the data received.

TOKEN(*data-area***)**
specifies the 8-character CICS-allocated token used to identify a report.

Restriction: You must specify the RESP or NOHANDLE option on the **EXEC CICS SPOOLREAD** command.

Conditions

Note: There are no default actions.

85 ALLOCERR
occurs in any of the following situations:

- Dynamic allocation has rejected a request to allocate an input data set.

  RESP2 gives the dynamic allocation response code that denotes this error. The first two characters are the information reason code (S99INFO), and the second two are the error reason code (S99ERROR), as defined in the *z/OS: MVS Programming: Authorized Assembler Services Guide, SA22-7608*.

20 ENDFILE
occurs in any of the following situations:

- All data for the current spool file being read has been retrieved. You should proceed to issue a SPOOLCLOSE command as soon as possible, to release the lock on the JES single thread, and to terminate current SYSOUT data set processing.

21 ILLOGIC
RESP2 values:

3 Invalid CLASS value specified.

16 INVREQ
RESP2 values:

4 Unsupported language.

8 Unsupported function.

12 Read attempt after end of file.

24 INTO missing.

40 Subsystem interface already enabled.

Note: Errors 1024 and over are internal, and should not occur. If one of these error codes is returned, contact your IBM support center.

22 LENGERR
occurs in any of the following situations:

- You provided insufficient buffer space to read your record, or you requested more than the maximum allowable buffer size of 32 760 bytes (the size of a CICS buffer used to read a record). If the buffer space is too small, it receives as much data as possible. The amount of data truncated is then placed in the RESP2 field. If the TOFLENGTH operand is specified, the actual length of the record is placed here.

  Note: In the event of a length error due to insufficient buffer space, the next record is not read until the error has been corrected and the current record reread.

  RESP2 indicates the amount of data truncated, or shows zero if the MAXFLENGTH field is greater than the maximum allowable buffer size 32 760 bytes.

80 NOSPOOL
RESP2 values:

4 No subsystem present.

8 Interface being disabled; CICS is quiescing.

12 Interface has been stopped.

42 NOSTG
occurs in any of the following situations:

- A GETMAIN has failed within the JES interface subtask (DFHPSPSS).

  RESP2 gives the GETMAIN register 15 return code.

13 NOTFND
RESP2 values:

4 No data sets could be located for retrieval for the specified external writer name.

19 NOTOPEN
RESP2 values:

8 Data set has not been opened.

12 Attempt to read an output file.

1024 Subtask OPEN macro failure.

88 SPOLBUSY
RESP2 values:

4 Interface already in use by another task.

8 Interface already in use by current task.

Also occurs (RESP2 not set) in any of the following situations:

- The JES/input single thread within the JES interface was not available.

89 SPOLERR
occurs in any of the following situations:

- The subsystem interface macro (IEFSSREQ) has failed. No input data set name was selected.

  RESP2 gives the 'IEFSSREQ' response code.

86 STRELERR
occurs in any of the following situations:

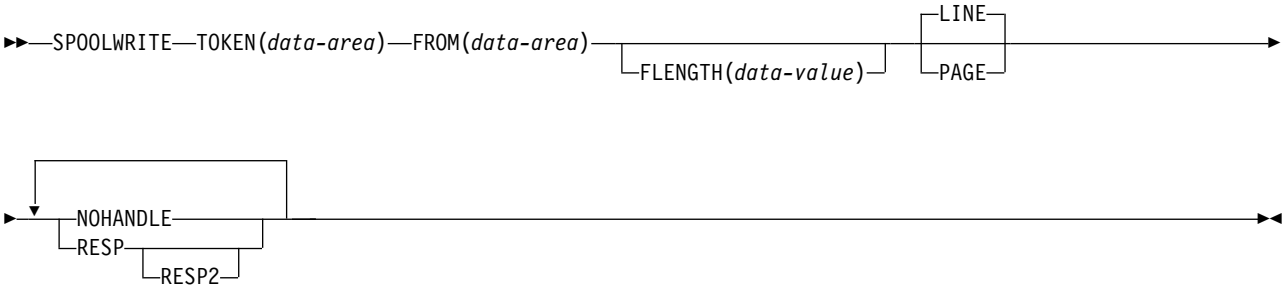
- A FREEMAIN has failed within the JES interface subtask (DFHPSPSS).

  RESP2 gives the FREEMAIN register 15 return code.

SPOOLWRITE

The SPOOLWRITE command writes data to a spool report.

SPOOLWRITE



Conditions: ALLOCERR, INVREQ, LENGERR, NOSPOOL, NOSTG, NOTOPEN, SPOLBUSY, SPOLERR, STRELERR

Options

FLENGTH(*data-value***)**
specifies the fullword binary variable that is to be set to the length of the data that is transferred. This is set by the user on output. It is optional and, if it is omitted, CICS uses the length of the data area.

FROM(*data-area***)**
specifies the data area from which to take the variable length data. The data itself is not altered in any way by CICS. FROM is a sender field.

LINE|PAGE
specifies the format of the data to be sent. The default action is LINE.

The PAGE option must be used to correctly format information for the advanced function printer (AFP) page printing devices. If a customer is

creating MIXED mode type data, that is LINE records and X'5A' (AFPDS or MODCA) pagemode records, the LINE or PAGE operand must match the type record being written to spool.

TOKEN(*data-area***)**
specifies the 8-character CICS-allocated token used to identify a report. It is a receiver on SPOOLOPEN and a sender on all other commands.

Restriction: You must specify the RESP or NOHANDLE option on the **EXEC CICS SPOOLWRITE** command.

Conditions

Note: There are no default actions.

85 ALLOCERR
occurs in any of the following situations:

- Dynamic allocation has rejected a request to allocate an input data set.

  RESP2 gives the dynamic allocation response code that denotes this error. The first two characters are the information reason code (S99INFO), and the second two are the error reason code (S99ERROR), as defined in the *z/OS: MVS Programming: Authorized Assembler Services Guide, SA22-7608*.

16 INVREQ
RESP2 values:

4 Unsupported language.

8 Unsupported function.

28 FROM missing.

40 Subsystem interface already enabled.

Note: Errors 1024 and over are internal, and should not occur. If one of these error codes is returned, contact your IBM support center.

22 LENGERR
occurs in any of the following situations:

- The value specified in the FLENGTH parameter on a SPOOLWRITE command is not in the valid range 1 to RECORDLENGTH value specified or defaulted at the SPOOLOPEN data set. If the buffer space is too small, it receives as much data as possible.

  RESP2 contains the difference between FLENGTH and RECORDLENGTH, or zero if FLENGTH is negative or greater than 32760.

80 NOSPOOL
RESP2 values:

4 No subsystem present.

8 Interface being disabled; CICS is quiescing.

12 Interface has been stopped.

42 NOSTG
occurs in any of the following situations:

- A GETMAIN has failed within the JES interface subtask (DFHPSPSS).

  RESP2 gives the GETMAIN register 15 return code.

19 NOTOPEN
RESP2 values:

8 Spool report has not been opened.

16 Attempt to write an input file.

1024 Subtask OPEN macro failure.

88 SPOLBUSY
RESP2 values:

4 Interface already in use by another task.

8 Interface already in use by current task.

Also occurs (RESP2 not set) in the following situation:

- The JES/input single thread within the JES interface was not available.

89 SPOLERR
occurs in the following situation:

- The subsystem interface macro (IEFSSREQ) has failed. No input data set name was selected.

  RESP2 gives the 'IEFSSREQ' response code.

86 STRELERR
occurs in the following situation:

- A FREEMAIN has failed within the JES interface subtask (DFHPSPSS).

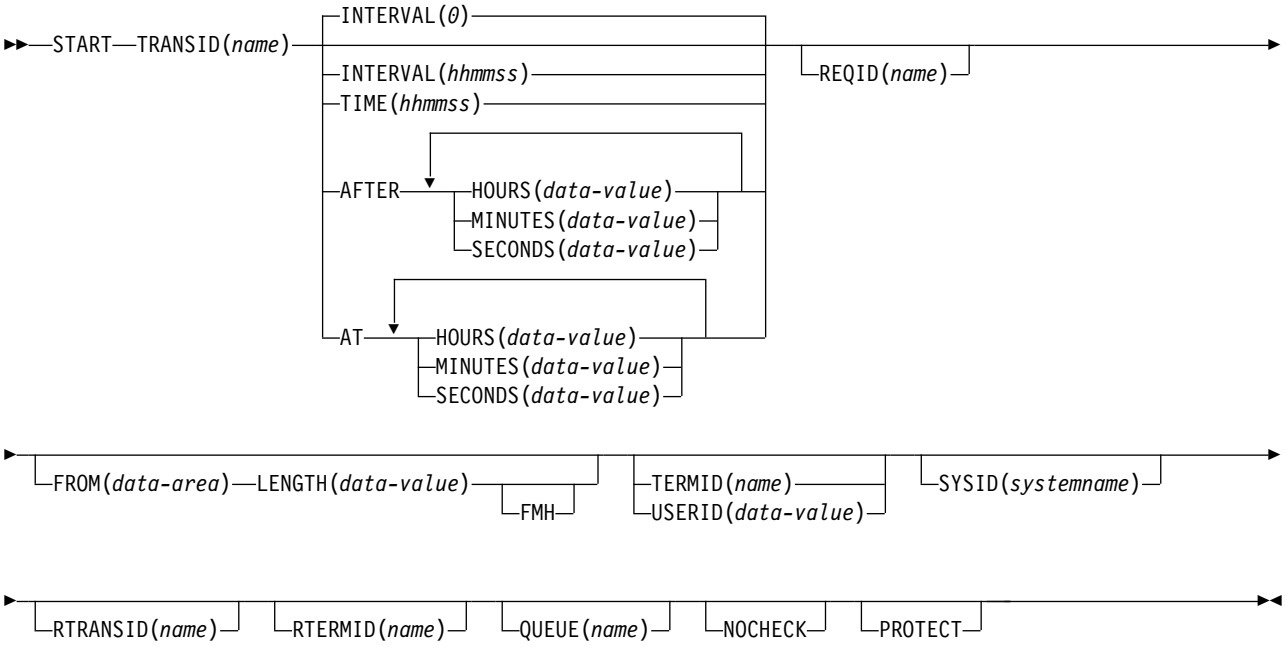
  RESP2 gives the FREEMAIN register 15 return code.

START

Start a task at a specified time.

See also:
- START ATTACH
- START BREXIT
- START CHANNEL

START



Conditions: INVREQ, IOERR, ISCINVREQ, LENGERR, NOTAUTH, RESUNAVAIL, SYSIDERR, TERMIDERR, TRANSIDERR, USERIDERR

Note: For dynamic transaction routing: If a START is later canceled by another task, or if the started transaction uses RETRIEVE WAIT, intertransaction affinities might be created that adversely affect the use of dynamic transaction routing. For more information, see Affinity .

Description

START starts a task, on a local or remote system, at a specified time. The time is specified by INTERVAL, AFTER, AT, or TIME. See the section about expiration times in Interval control.

The starting task can pass data to the started task. The starting task can also specify a terminal to be used by the started task as its principal facility.

The default is INTERVAL(0), but for C the default is AFTER HOURS(0) MINUTES(0) SECONDS(0).

CEDF is an exception to the START command and is not valid as a TRANSID name. Therefore, do not attempt to start CEDF in this way.

Important: There is a 6-hours rule, if the START gets triggered at any time within 6 hours after the time specified on the START, it executes immediately. This happens regardless of whether the previous 6 hours includes a midnight. To illustrate here are some examples:

- `EXEC CICS START TIME(123000)`

  This command, issued at 05:00 or 07:00 on Monday, expires at 12:30 on the same day.

- `EXEC CICS START TIME(020000)`

  This command, issued at 05:00 or 07:00 on Monday expires immediately because the specified time is within the preceding 6 hours.

For more information, see Expiration times.

You can use the RTRANSID, RTERMID, and QUEUE options to pass further data to the started task. These options can contain arbitrary data values with meanings that depend on what is specified in the started and starting tasks. One possible way of using them is in the following situation. One task can start a second task, passing it a transaction name and a terminal name to be used when the second task starts a third task. The first task can also pass the name of a queue to be accessed by the second task.

If you are using an IPIC connection, the maximum length for the FROM data area is 32,500 bytes. This limit allows for the 32,500 byte FROM data area and space for headers.

START with TERMID specified does not propagate the origin data record (ODR), so tasks are always started at a new point of origin.

The following constraints must be satisfied before the transaction to be run can be started:

- The specified interval must have elapsed or the specified expiration time must be reached. For more information, see Interval control. Specify the INTERVAL or AFTER options when a transaction is to be run on a remote system, to avoid complications when the local, and remote systems are in different time zones.
- If you specify the TERMID option, the named terminal must exist and be available. If the named terminal does not exist when the time interval expires, the START is discarded.
- If you specify the PROTECT option, the starting task must take a successful sync point. This option, coupled to extensions to system tables, reduces the exposure to lost or duplicated data caused by failure of a starting task.
- If the transaction to be run is on a remote system, the format of the data must be declared to be the same as the data at the local system. Use the RDO options DATASTREAM and RECORDFORMAT. For CICS to CICS, these options are always the default values. For CICS-to-IMS/VS, make sure that you specify the correct values.

Running a START command that names a transaction in the local system cancels any outstanding POST commands run by the starting task.

You can queue START commands by specifying the LOCALQ option on the RDO TRANSACTION resource definition, as described in TRANSACTION attributes.

Passing data by interval control

If data is to be passed by interval control (by using the FROM option), it is queued on a temporary storage queue. Use the REQID option to specify the name of the temporary storage queue to be used. This identifier might be recoverable (in temporary storage terms) or unrecoverable. For more information on how to define recoverable temporary storage queues, see TSMODEL resources.

If you also specify the PROTECT option, you must define the temporary storage queue identified by the REQID option as recoverable. If you do not specify the

PROTECT option, do not define the temporary storage queue as recoverable. Unpredictable results can occur if you do not follow these rules. See Recovery of START requests.

If you specify the FROM and not the REQID option, a default 'DF' prefix temporary storage queue is used. The same rules apply as previously listed; specify the PROTECT option only if you define the 'DF' prefix temporary storage queues as recoverable.

A START command with REQID, issued from a task that was itself initiated by a START with the same REQID, returns an AEIQ abend (IOERR condition) if the FROM data for the task has not been read by a RETRIEVE.

You also receive this error if more than one START command with the same REQID is issued by a task or tasks in the same CICS system. CICS TS regions always reject with an IOERR any START commands that specify a duplicate REQID.

Started tasks without data run without a facility address. Started tasks with data run with a facility address of an ICE until the data is retrieved.

If an ICRX is used, it is saved across restarts. If the start request is subsequently cancelled, the ICRX is deleted.

Error checking and performance considerations

The NOCHECK option specifies that no response (to running of the START command) is expected by the starting transaction. For START commands that name tasks to be started on a local system, error conditions are returned; error conditions are not returned for tasks to be started on a remote system. The NOCHECK option allows CICS to improve performance when the START command must be shipped to a remote system; it is also a prerequisite if the shipping of the START command is queued pending the establishing of links to the remote system.

Starting tasks without terminals

If the task to be started is not associated with a terminal, each START command starts a separate task, regardless of whether data is passed to the started task. The following examples show how to start a specified task, not associated with a terminal, in one hour:

```
EXEC CICS START
     TRANSID('TRNL')
     INTERVAL(10000)
     REQID('NONGL')
⋮
EXEC CICS START
     TRANSID('TRNL')
     AFTER HOURS(1)
     REQID('NONGL')
⋮
```

Starting tasks with terminals but without data

Only one task is started if several START commands, each specifying the same transaction and terminal, expire at the same time or before the terminal is available.

The following examples show how to request initiation of a task associated with a terminal. Because no request identifier is specified in these examples, CICS assigns one and returns it to the application program in the EIBREQID field of the EXEC interface block.

```
EXEC CICS START
     TRANSID('TRN1')
     TIME(185000)
     TERMID('STA5')
⋮
EXEC CICS START
     TRANSID('TRN1')
     AT HOURS(18) MINUTES(50)
     TERMID('STA5')
⋮
```

Starting tasks with terminals and data

Data is passed to a started task if one or more of the FROM, RTRANSID, RTERMID, and QUEUE options is specified. Such data is accessed by the started task by using a RETRIEVE command.

It is possible to pass many data records to a new task by issuing several START commands, each specifying the same transaction, and terminal.

Running of the first START command ultimately causes the new task to be started and allows it to retrieve the data specified on the command. The new task is also able to retrieve data specified on subsequent START commands that expire before the new task is ended. If the transaction has been defined as restartable (by defining the transaction as using the RDO option RESTART(YES)) and such data has not been retrieved before the new task is ended, another new task is started and can retrieve the outstanding data.

If the transaction abends and has not been defined as restartable, no new task is initiated and the data is discarded.

The following examples show how to start a task associated with a terminal and to pass data to it:

```
EXEC CICS START
     TRANSID('TRN2')
     TIME(173000)
     TERMID('STA3')
     REQID(DATAREC)
     FROM(DATAFLD)
     LENGTH(100)
⋮
EXEC CICS START
     TRANSID('TRN2')
     AT HOURS(17) MINUTES(30)
     TERMID('STA3')
     REQID(DATAREC)
     FROM(DATAFLD)
     LENGTH(100)
⋮
```

When you use the C language, use the AFTER/AT HOURS, MINUTES, and SECONDS options because C does not provide a packed decimal data type. You can use INTERVAL or TIME, but if the value specified is *not* an integer constant, the application is responsible for ensuring that the value passed to CICS is in packed decimal format.

Dynamically routed transactions started by START commands

Some transactions started by a subset of START commands can be dynamically routed to a remote region. For general information about dynamic transaction routing, and specific information about which transactions started by START commands are eligible for dynamic routing, see Routing transactions invoked by START commands.

Note: You cannot use identity propagation with a START command that is shipped to a remote region across a LU61 or LU62 connection. For this type of connection, the ICRX will not be shipped and the identity information is lost. Identity propagation is also not available for delayed dynamically routed START commands.

START failures without exception conditions

In the following circumstances, a START command runs without error, but the started task never takes place:

- When the transaction or its initial program is unavailable at the time CICS attempts to create the task.
- When the START specifies a terminal and an expiration time, and the terminal is not defined (and cannot be located by the XICTENF or XALTENF exits) at expiration time.
- When the START specifies a terminal that is not defined (and cannot be located by the XICTENF or XALTENF exits) at the time CICS attempts to create the task.

These exposures result from the delay between the running of the START command and the time of task creation. Even when the START is immediate, CICS can delay creating the task, either because the required terminal is not free or because of other system constraints.

You can use INQUIRE commands to ensure that the transaction and program are enabled at the time of the START command, but either might become unavailable before task creation.

You get a TERMIDERR condition if the requested terminal does not exist at the time of the START, but if it is deleted later, as occurs if the user logs off, your START request is discarded with the terminal definition.

Application context propagation

Application context propagation is a process that passes application context data from a task that issues a START command to a task that runs the started transaction. The data is used for monitoring and measuring application resource usage. Both the initial and current application contexts are passed between applications. Application context data *is not* propagated with the following START commands:

- A local **EXEC CICS START** command generated from a transient data (TD) trigger.
- A local asynchronous **EXEC CICS START** command received from an event processing (EP) adapter.
- An **EXEC CICS START** command that specifies a **TERMID** or a **USERID** parameter value.

For more information, see Application context.

Options

AFTER
Specifies the interval of time that is to elapse before the new task is started.

Enter the time under AFTER and AT in two ways:

1. A combination of at least two of HOURS(0 - 99), MINUTES(0 - 59), and SECONDS(0 - 59). HOURS(1) SECONDS(3) means one hour and three seconds (the minutes default to zero).
2. As one of HOURS(0 - 99), MINUTES(0 - 5999), or SECONDS(0 - 359999). HOURS(1) means one hour. MINUTES(62) means one hour and two minutes. SECONDS(3723) means one hour, two minutes, and three seconds.

AT Specifies the time at which the new task is to be started. For the ways to enter the time, see the AFTER option.

FMH
Specifies that the user data to be passed to the started task contains function management headers. FMH is not valid for LUTYPE2 or LUTYPE3 terminals.

FROM(*data-area***)**
Specifies the data to be stored for a task that is to be started at some future time.

HOURS(*data-value***)**
Specifies a fullword binary value in the range 0 - 99. HOURS is a suboption of the AFTER and AT options. For its use and meaning, see the AFTER option.

INTERVAL(*hhmmss***)**
Specifies the expiration time as an interval of time that is to elapse from the time at which the START command is issued. The *mm* and *ss* are each in the range 0 - 59. The time specified is added to the current clock time by CICS when the command is run, to calculate the expiration time.

LENGTH(*data-value***)**
Specifies a halfword binary data value that is the length of the data to be stored for the new task. See “LENGTH options in CICS commands” on page 10 for guidance on limits when setting the LENGTH option.

MINUTES(*data-value***)**
Specifies as a fullword binary value the number of minutes for use with AFTER or AT. The value must be in the range 0 through 59 if HOURS or SECONDS is also specified, or in the range 0 through 5999 otherwise. MINUTES is a suboption of the AFTER and AT options. For its use and meaning, see the AFTER option.

NOCHECK
Specifies that, for a remote system, CICS improves performance of the START command by providing less error checking and slightly less function. For more information, see Improving performance of intersystem START requests

PROTECT
Specifies that the new task is not started until the starting task has taken a sync point. If the starting task abends before the sync point is taken, the request to start the new task is canceled. If the REQID option is also specified, the request identifier must be a name defined as recoverable to temporary storage. If the started transaction is remote, PROTECT specifies that it must not be scheduled until the local transaction has successfully completed a sync point. For more information about the PROTECT option with remote transactions, see Improving performance of intersystem START requests

QUEUE(*name***)**
Specifies a name (1 - 8 characters) that is passed to the started task. If this name represents a temporary storage queue, the queue must be local to the started task. The contents of the queue are not passed.

If you are also specifying REQID, make sure that the name of the REQID and the name of the QUEUE are not the same.

REQID(*name***)**
Specifies a name (1 - 8 characters), which must be unique, to identify a command. You can use this option when another task is to be provided with the capability of canceling an unexpired command.

If you omit this option, CICS generates a unique request identifier in the EIBREQID field of the EXEC interface block, unless you specify the NOCHECK option, in which case field EIBREQID is set to nulls and cannot be used later to cancel the START command.

If you include any of the data options (FROM, RTERMID, RTRANSID, or QUEUE), the data is stored in a TS queue by using the REQID name specified (or CICS generated) as the identifier. The queue record thus identified must be local to the CICS system where the START command is processed. The START command is processed on the system identified by the SYSID option or, if the SYSID option is omitted, on the system associated with the TRANSID option.

RTERMID(*name***)**
Specifies a value (1 - 4 characters), for example a terminal name, that can be retrieved when the transaction, specified in the TRANSID option in the START command, is started.

When retrieved, the value can be used in the TERMID option of a subsequent START command.

RTRANSID(*name***)**
Specifies a value (1 - 4 characters), for example a transaction name, that can be retrieved when the transaction, specified in the TRANSID option in the START command, is started.

When retrieved, the value can be used in the TRANSID option of a subsequent START command.

SECONDS(*data-value***)**
Specifies a fullword binary value in the range 0 - 59, when HOURS or MINUTES are also specified, or 0 - 359999 when SECONDS is the only option specified. SECONDS is a suboption of the AFTER and AT options. For its use and meaning, see the AFTER option.

SYSID(*systemname***)**
Specifies the name of the system to which the request is directed.

TERMID(*name***)**
Specifies the symbolic identifier (1 - 4 alphanumeric characters) of the principal facility associated with a transaction to be started as a result of a START command. This principal facility can be either a terminal (the usual case) or an APPC session. Where an APPC session is specified, the connection (or modeset) name is used instead of a terminal identifier. This option is required when the transaction to be started must communicate with a terminal; it should be omitted otherwise.

Define the terminal identifier as either a local or a remote terminal on the system in which the START command is run when the start of the transaction takes effect.

The TERMID option is used in previous hop data that is collected. For more information about using the TERMID option with previous hop data, see Previous hop data characteristics.

Do not use the TERMID option if you are using identity propagation.

TIME(*hhmmss***)**
Specifies the time when a new task is started.

When using the C language, use the AFTER/AT HOURS, MINUTES, and SECONDS options because C does not provide a packed decimal data type. You can use TIME, but if the value specified is *not* an integer constant, the application is responsible for ensuring that the value passed to CICS is in packed decimal format.

TRANSID(*name***)**
Specifies the symbolic identifier (1 - 4 characters) of the transaction to be run by a task started as the result of a START command.

If you specify SYSID and name a remote system, the transaction is assumed to be on that system whether the transaction is defined as remote. Otherwise, the transaction resource definition is used to find out whether the transaction is on a local or a remote system.

The TRANSID option is used in previous hop data that is collected. For more information, see Previous hop data characteristics.

USERID(*data-value***)**
Specifies the user ID under whose authority the started transaction is to run, if the started transaction is not associated with a terminal (that is, when TERMID is not specified). This user ID is referred to as *userid1*.

If you omit both TERMID and USERID, CICS uses instead the user ID under which the transaction that issues the START command is running. This user ID is referred to as *userid2*.

By using either *userid1* or *userid2*, CICS ensures that a started transaction always runs under a valid user ID, which must be authorized to all the resources referenced by the started transaction.

CICS performs a surrogate security check against *userid2* to verify that this user is authorized to *userid1*. If *userid2* is not authorized, CICS returns a NOTAUTH condition.

If you are using identity propagation, and your task has a distributed user ID associated with its security context, this information is not propagated to tasks that are started with the USERID option.

Conditions

16 INVREQ

RESP2 values:

4 The value specified in HOURS, for AFTER or AT options, or the *hh* value specified for INTERVAL, is out of range.

5 The value specified in MINUTES, for AFTER or AT options, or the *mm* value specified for INTERVAL, is out of range.

6 The value specified in SECONDS, for AFTER or AT options, or the *ss* value specified for INTERVAL, is out of range.

17 The transaction that has been started by the START operation is not shutdown-enabled, and the CICS region is in the process of shutting down.

18 A USERID is specified and the CICS external security manager interface is not initialized.

200 The START command was issued by a distributed program link (DPL) server program and used the TERMID option, the value of which matched the ID of the intersystem session. Where TERMID is equal to EIBTRMID (that is, where the issuing task's principal facility is a session rather than a terminal), START TERMID is not in the permitted subset of commands available to DPL server programs.

INVREQ also occurs (RESP2 not set) in any of the following situations:

- The START command is not valid for processing by CICS.
- Values specified in the INTERVAL option are out of range.

Default action: end the task abnormally.

17 IOERR

Occurs in any of the following situations:

- An input/output error occurred during a START operation.
- A START operation attempts to write to a temporary storage queue when the DFHTEMP data set is full.
- A START operation uses a REQID name that exists. This condition occurs only when the FROM option is also used.

Default action: end the task abnormally.

54 ISCINVREQ

Occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: end the task abnormally.

22 LENGERR
Occurs if LENGTH is not greater than zero.

Default action: end the task abnormally.

70 NOTAUTH
RESP2 values:

7 A resource security check fails on TRANSID *(name)*.

9 A surrogate user security check fails on USERID *(name)*.

The security access capabilities of the transaction that issued the command do not allow the command to be performed with the value specified in the USERID option. The security access capabilities of the transaction have been established by the external security manager according to user security, and whether link security or the execution diagnostic facility (EDF) have been in use.

Default action: end the task abnormally.

121 RESUNAVAIL
RESP2 values:

121 A resource required by the transaction to be started is unavailable on the target region. The RESUNAVAIL condition applies only to dynamically-routed, non-terminal-related EXEC CICS START requests.

RESUNAVAIL is returned on the EXEC CICS START command run by the mirror in the target region, if an XICERES global user exit program indicates that a required resource is unavailable on the target region. It is not returned to the application.

Default action: reinvoke the distributed routing program for route selection failure.

53 SYSIDERR
Occurs when the SYSID option specifies a name that is neither the local system nor a remote system (made known to CICS by defining a CONNECTION). SYSIDERR also occurs when the link to the remote system is known but unavailable.

The following error is indicated by a RESP2 value:

1 The dynamic routing program rejected the START request.

Default action: end the task abnormally.

11 TERMIDERR
Occurs if the terminal identifier in a START command is not defined to CICS. This condition can be produced if you specify the name of a connection that is not an ISC or MRO connection.

Default action: end the task abnormally.

28 TRANSIDERR
Occurs if the transaction identifier specified in a START command is not defined to CICS.

Default action: end the task abnormally.

69 USERIDERR
RESP2 values:

8 The specified USERID is not known to the external security manager.

10 The external security manager is in a state such that CICS cannot determine whether a specified USERID is valid.

19 The specified USERID is revoked.

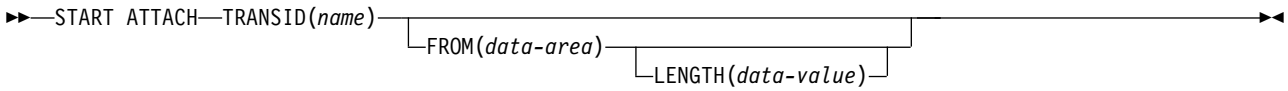
Default action: end the task abnormally.

START ATTACH

Start a task immediately.

See also:
- "START" on page 561
- "START BREXIT" on page 573
- "START CHANNEL" on page 576

START ATTACH



Conditions: INVREQ, LENGERR, NOTAUTH, TRANSIDERR

Description

START ATTACH starts a non-terminal task immediately in the local CICS region.

The attached task has a STARTCODE of U and cannot be canceled, so EIBREQID is set to nulls.

ATTACH allows a START issued in a PLTPI program to take effect before initialization completes.

The starting task can pass data to the started task by using the FROM option.

START ATTACH does not propagate the origin data record (ODR), so tasks are always started at a new point of origin

Starting tasks with data

If data is to be passed, it is not written to a temporary storage queue; only its address is passed.

The attached task retrieves data in the normal way. The task that issued the START must ensure that the data is valid when it is retrieved, either by synchronizing its execution with the attached task, or by placing the data in shared storage.

Each **START ATTACH** command results in a separate task started, optionally with data passed to the started task. The following example shows how to start a specified task, and pass data to it:

```
EXEC CICS START ATTACH
          TRANSID('TRNL')
          FROM(DATAFLD)
          LENGTH(100)
⋮
```

Options

FROM(*data-area***)**
Specifies the data to be passed to a started task.

LENGTH(*data-value***)**
Specifies a halfword binary data value that is the length of the data to be passed to a started task.

TRANSID(*name***)**
Specifies the symbolic identifier (1–4 characters) of the transaction to be started by a task started as the result of a START ATTACH command.

The TRANSID option is used in previous hop data that is collected. For more information, see Previous hop data characteristics.

Conditions

16 INVREQ
RESP2 values:

11 An attempt was made to route a START ATTACH request.

12 A START ATTACH request failed.

Default action: terminate the task abnormally.

22 LENGERR
Occurs if LENGTH is not greater than zero.

Default action: terminate the task abnormally.

70 NOTAUTH
RESP2 values:

7 A resource security check fails on TRANSID (name).

Default action: terminate the task abnormally.

28 TRANSIDERR
Occurs if the transaction identifier specified in a START command is not defined to CICS.

RESP2 values:

11 The specified transaction is defined as remote.

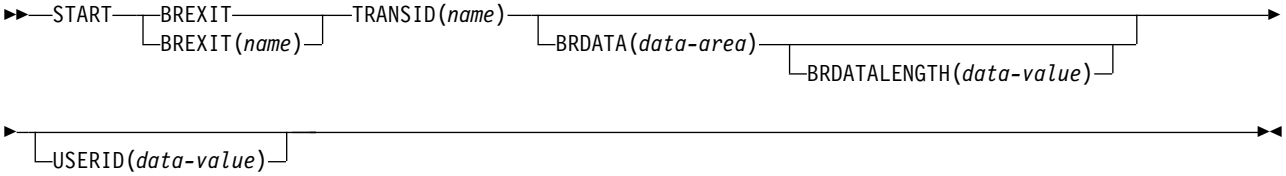
Default action: terminate the task abnormally.

START BREXIT

Start task in the 3270 bridge environment and associate it with the named bridge exit.

See also:
- "START" on page 561
- "START ATTACH" on page 572
- "START CHANNEL" on page 576

START BREXIT



Conditions: INVREQ, LENGERR, NOTAUTH, PGMIDERR, TRANSIDERR, USERIDERR

Description

START BREXIT starts a task immediately in the local CICS region, and initializes the specified transaction (TRANSID) and bridge exit (BREXIT). In the 3270 bridge environment, all 3270 terminal requests issued by the transaction specified by TRANSID, are intercepted and passed to the user-replaceable program (the bridge exit) specified by BREXIT.

The bridge exit (BREXIT) emulates the 3270 interface by passing the terminal requests to a client application that might be executing inside or outside of CICS.

See Introduction to the 3270 bridge for more information about the 3270 bridge and its interfaces.

The attached task cannot be canceled; its STARTCODE is defined by the bridge exit.

Options

BREXIT(*name***)**
Specifies the name (1-8 characters) of the bridge exit to be associated with the started task. If no name is specified, the value of BREXIT on the TRANSACTION resource definition for TRANSID is used.

BRDATA(*data-area***)**
Specifies the data to be passed to the bridge exit specified by BREXIT when the task is started.

BRDATALENGTH(*data-value***)**
Specifies a fullword binary data value that is the length of the BRDATA to be passed to the bridge exit specified by BREXIT when the task is started.

TRANSID(*name***)**
Specifies the symbolic identifier (1–4 characters) of the transaction to be executed by a task started as the result of a START BREXIT command. The transaction is started in the 3270 bridge environment, and executes in association with the bridge exit specified in BREXIT.

The TRANSID option is used in previous hop data that is collected. See Association data for more information about using the TRANSID option with previous hop data.

USERID(*data-value***)**
Specifies the user ID under whose authority the started transaction is to run.

Conditions

16 INVREQ

RESP2 values:

11 An attempt was made to route a START BREXIT request.

12 A START BREXIT request has failed.

18 A USERID option is specified and the CICS external security manager interface is not initialized.

Default action: terminate the task abnormally.

22 LENGERR

Occurs if BRDATALENGTH is not greater than zero.

Default action: terminate the task abnormally.

70 NOTAUTH

RESP2 values:

7 A resource security check fails on TRANSID (name).

9 A surrogate user security check fails on USERID (name). The security access capabilities of the transaction that issued the command do not allow the command to be performed with the value specified in the USERID option.

Default action: terminate the task abnormally.

27 PGMIDERR

Occurs if no name is supplied by the BREXIT option and the transaction definition for TRANSID does not provide a default BREXIT name.

Default action: terminate the task abnormally.

28 TRANSIDERR

Occurs if the TRANSID specified in a START BREXIT command has not been defined to CICS.

RESP2 values:

11 The specified transaction is defined as remote.

Default action: terminate the task abnormally.

69 USERIDERR

RESP2 values:

8 The specified user ID is not known to the external security manager.

10 The external security manager is in a state such that CICS cannot determine whether a specified user ID is valid.

Default action: terminate the task abnormally.

Passing data to the bridge exit

Data can be passed to the bridge exit by using the BRDATA and BRDATALENGTH options.

The following example shows how to start a specified task, in the 3270 bridge environment and pass data to its bridge exit:

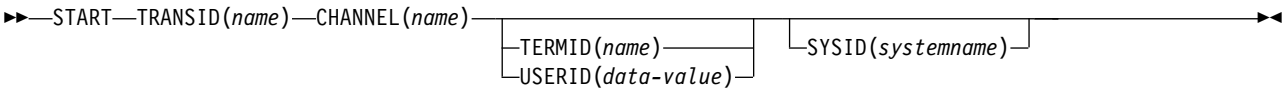
```
EXEC CICS START BREXIT('DFH0CBRE')
          TRANSID('TRNL')
          BRDATA(BRSD)
          BRDATALENGTH(72)
⋮
```

START CHANNEL

Start a task, passing it a channel.

See also:
- "START" on page 561
- "START ATTACH" on page 572
- "START BREXIT" on page 573

START CHANNEL



Conditions: CHANNELERR, INVREQ, ISCINVREQ, NOTAUTH, RESUNAVAIL, SYSIDERR, TERMIDERR, TRANSIDERR, USERIDERR

Description

START CHANNEL starts a task on a local or remote system, passing it a channel.

Typically, the starting task uses the channel to pass data to the started task (although in some circumstances the channel can be empty; see the description of the CHANNEL option). The starting task can also specify a terminal to be used by the started task as its principal facility.

For example, the started task can do the following:

1. Use an **ASSIGN CHANNEL** command to discover the name of the channel that has been passed to it.
2. Use **STARTBROWSE CONTAINER CHANNEL** and **GETNEXT CONTAINER** commands to browse the containers in the channel.
3. Use **GET CONTAINER CHANNEL** or **GET64 CONTAINER** commands to access the data in the containers.

The following constraints must be satisfied before the transaction to be run can be started:

- If you specify the TERMID option, the named terminal must exist and be available. If the named terminal does not exist, the START is discarded.
- START CHANNEL does not support IMS; that is, you cannot use START CHANNEL to start a transaction on a remote IMS system.

Each START CHANNEL command results in a separate task being started.

START with TERMID specified does not propagate the origin data record (ODR), so tasks are always started at a new point of origin.

Dynamically routed transactions started by START commands

Some transactions started by a subset of START commands can be dynamically routed to a remote region. For general information about dynamic transaction routing, and specific information about which transactions started by START commands are eligible for dynamic routing, see Routing transactions invoked by START commands.

START failures without exception conditions

In the following circumstances, a START command runs without error, but the started task never takes place:

- When the transaction or its initial program is unavailable at the time CICS attempts to create the task.
- When the START specifies a terminal that is not defined (and cannot be located by the XICTENF or XALTENF exits) at the time CICS attempts to create the task.
- You get a TERMIDERR condition if the requested terminal does not exist at the time of the START. However, if the terminal becomes unavailable later, as occurs if the user logs off, your START request is discarded, and no TERMIDERR occurs.

These exposures result from the delay between running the START command and the time of task creation. Even on a START CHANNEL request, when the START is always immediate, CICS can delay creating the task, either because the required terminal is not free or because of other system constraints.

You can use INQUIRE commands to ensure that the transaction and program are enabled at the time of the START command, but either can become unavailable before task creation.

Options

CHANNEL(*name*)

Specifies the name (1 - 16 characters) of a channel that is to be made available to the started task. The acceptable characters are A-Z a-z 0-9 $ @ # ./-_ % & ? ! : | " = ¬ , ; < >. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters. If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

Channel names are always in EBCDIC. The set of allowed characters for channel names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be shipped between regions, it is advisable to restrict the characters used to name them to A-Z a-z 0-9 & : = , ; < > . - and _.

You can specify the channel name `DFHTRANSACTION` to use a transaction channel. A transaction channel does not go out of scope when the link level changes: it is always accessible in the transaction. For more information, see Channels and containers.

The program that issues the START command can do one of the following:

- Create the channel before issuing the START command, by using one or more **PUT CONTAINER CHANNEL** or **PUT64 CONTAINER** commands.
- Specify its current channel, by name.
- Name a channel that does not currently exist. A new empty channel is created.

The started task is given a *copy* of the channel's containers (and the data that they contain). The copy is made when the START command is issued.

Note: All EXEC CICS START requests that specify a channel cannot be deferred by specifying **INTERVAL**, **AT**, **FOR** or **UNTIL** as this is not supported.

SYSID(*systemname***)**
Specifies the name of the system to which the request is directed.

TERMID(*name***)**
Specifies the symbolic identifier (1 - 4 alphanumeric characters) of the principal facility associated with a transaction to be started as a result of a START command. This principal facility can be either a terminal (the usual case) or an APPC session. Where an APPC session is specified, the connection (or modeset) name is used instead of a terminal identifier. This option is required when the transaction to be started must communicate with a terminal; it should be omitted otherwise.

You must define the terminal identifier as either a local or a remote terminal on the system in which the START command is run.

The TERMID option is used in previous hop data that is collected. For more information, see Previous hop data characteristics.

Do not use the TERMID option if you are using identity propagation.

TRANSID(*name***)**
Specifies the symbolic identifier (1 - 4 characters) of the transaction to be run by a task started as the result of a START command.

If you specify the SYSID, and name a remote system, the transaction is assumed to be on that system irrespective of whether the transaction is defined as remote. Otherwise, the transaction definition is used to find out whether the transaction is on a local or a remote system.

The TRANSID option is used in previous hop data that is collected. For more information, see Previous hop data characteristics.

USERID(*data-value***)**
Specifies the user ID under whose authority the started transaction is to run, if the started transaction is not associated with a terminal (that is, when TERMID is not specified). This user ID is referred to as *userid1*.

If you omit both TERMID and USERID, CICS uses instead the user ID under which the transaction that issues the START command is running. This user ID is referred to as *userid2*.

By using either *userid1* or *userid2* CICS ensures that a started transaction always runs under a valid user ID, which must be authorized to all the resources referenced by the started transaction.

CICS performs a surrogate security check against *userid2* to verify that this user is authorized to *userid1*. If *userid2* is not authorized, CICS returns a NOTAUTH condition. The surrogate check is not done here if USERID is omitted.

Do not use the USERID option if you are using identity propagation.

Conditions

122 CHANNELERR

RESP2 values:

1 The channel specified on the CHANNEL option contains an incorrect character or combination of characters.

16 INVREQ

RESP2 values:

9 The options specified on the command are incompatible.

17 The STARTed transaction is not shutdown-enabled, and the CICS region is in the process of shutting down.

18 A USERID is specified and the CICS external security manager interface is not initialized.

200 The START command was issued by a distributed program link (DPL) server program and used the TERMID option, the value of which matched the ID of the intersystem session. Where TERMID is equal to EIBTRMID (that is, where the issuing task's principal facility is a session rather than a terminal), START TERMID is not in the permitted subset of commands available to DPL server programs.

INVREQ also occurs (RESP2 not set) if the START command is not valid for processing by CICS.

Default action: end the task abnormally.

54 ISCINVREQ

Occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: end the task abnormally.

70 NOTAUTH

RESP2 values:

7 A resource security check fails on TRANSID (name).

9 A surrogate user security check fails on USERID (name).

The security access capabilities of the transaction that issued the command do not allow the command to be performed with the value specified in the USERID option. The security access capabilities of the transaction have been established by the external security manager according to user security, and whether link security or the execution diagnostic facility (EDF) have been in use.

Default action: end the task abnormally.

121 RESUNAVAIL

RESP2 values:

121 A resource required by the transaction to be started is unavailable on the target region. The RESUNAVAIL condition applies only to *dynamically-routed, non-terminal-related* EXEC CICS START requests.

RESUNAVAIL is returned on the EXEC CICS START command *run by the mirror in the target region*, if an XICERES global user exit program indicates that a required resource is unavailable on the target region. It is not returned to the application.

Default action: reinvoke the distributed routing program for route selection failure.

53 SYSIDERR
Occurs in all of the following cases:

- The SYSID option specifies a name that is neither the local system nor a remote system (made known to CICS by defining a CONNECTION).
- The link to the remote system is known but unavailable.

In all the previous cases, the nature of the error is indicated by the second byte of the EIBRCODE.

The following errors are indicated by RESP2 values:

1 The dynamic routing program rejected the START request.

2 The CHANNEL option was used and the START request was shipped or routed to a remote system which does not support it. (MRO connections only).

Default action: end the task abnormally.

20 The CHANNEL option is specified and the START request is to be shipped over an LUTYPE61 connection. START CHANNEL requests cannot be shipped over LUTYPE61 connections.

11 TERMIDERR
The terminal identifier in a START command is not defined to CICS.

Default action: end the task abnormally.

28 TRANSIDERR
The transaction identifier specified in a START command is not defined to CICS.

Default action: end the task abnormally.

69 USERIDERR
RESP2 values:

8 The specified USERID is not known to the external security manager.

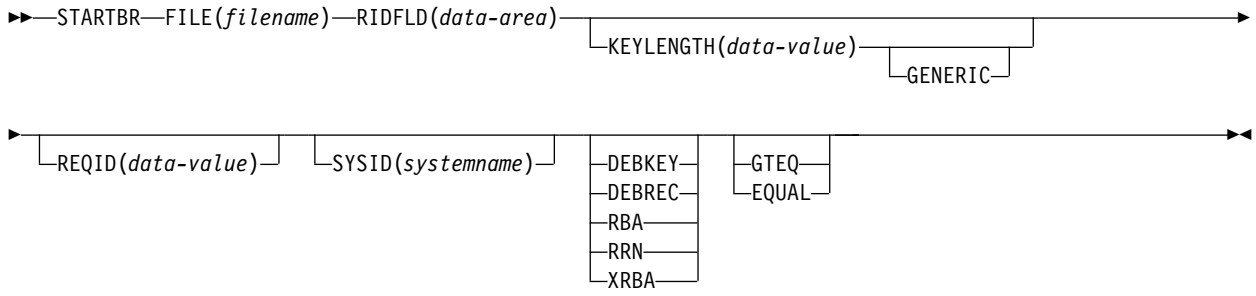
10 The external security manager is in a state such that CICS cannot determine whether a specified USERID is valid.

Default action: end the task abnormally.

STARTBR

Start browse of a file.

STARTBR



Conditions: DISABLED, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, LOADING, NOTAUTH, NOTFND, NOTOPEN, SYSIDERR

This command is threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over an IPIC connection to a remote CICS region.
- Defined as either local VSAM or RLS.

This command is not threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over a non-IPIC connection.
- Defined as a shared data table, coupling facility data table, or BDAM file.

Description

STARTBR specifies the record in a file, or in a data table, on a local or a remote system, where you want the browse to start. No records are read until a READNEXT command (or, for VSAM and tables, a READPREV command) is executed.

The following browse operations are possible. A direct browse is a browse of the base data set by using the primary key.

- A direct browse of a key sequenced data set (KSDS or data-table) by record key.
- A direct browse of an entry sequenced data set (ESDS) by relative byte address (RBA).
- A direct browse of a relative record data set (RRDS) by relative record number (RRN).
- A browse of a key sequenced data set (KSDS) using an alternate index path.
- A browse of an entry sequenced data set (ESDS) using an alternate index path. In this case, an ESDS is browsed by key in the same way as a KSDS. Some of the options that are not valid for a direct ESDS browse are valid for an alternate index browse.
- A browse of a KSDS by RBA.

The options specified on the STARTBR command define the characteristics that apply throughout the subsequent browse operation. Specifically, if GENERIC or GTEQ are specified, they are used not only when determining the starting point of the browse, but also whenever the value of RIDFLD is changed before issuing a READNEXT command.

If you specify the RBA option, it applies to every READNEXT or READPREV command in the browse, and causes CICS to return the relative byte address of each retrieved record.

None of these options can be changed during a browse, except by means of the RESETBR command.

If a STARTBR request specifies the precise key at which the browse is to start (that is, it specifies a full key and the EQUAL keyword) the record returned on the following READNEXT (or READPREV) may not be the same as the record specified by the STARTBR for a file opened in VSAM NSR or RLS mode. This can occur because the initial record specified on the STARTBR command can be deleted by another transaction in between the STARTBR completing and a READNEXT or READPREV being issued. In VSAM LSR mode, the initial record cannot be deleted between the STARTBR and the READNEXT.

A browse can end by using the ENDBR, SYNCPOINT, or SYNCPOINT ROLLBACK commands. Also an implicit sync point at the end of task ends the browse.

Options

DEBKEY
(blocked BDAM) specifies that deblocking is to occur by key. If neither DEBREC nor DEBKEY is specified, deblocking does not occur.

DEBREC
(blocked BDAM) specifies that deblocking is to occur by relative record (relative to zero). If neither DEBREC nor DEBKEY is specified, deblocking does not occur.

EQUAL
(VSAM and data table) specifies that the search is satisfied only by a record having the same key (complete or generic) as that specified in the RIDFLD option.

This option is the default field for a direct ESDS browse.

FILE(*filename***)**
specifies the name of the file to be accessed.

If SYSID is specified, the data set to which this file refers is assumed to be on a remote system irrespective of whether the name is defined to CICS. Otherwise, the resource definition is used to find out whether the data set is on a local or a remote system.

GENERIC
(VSAM KSDS, path or data table) specifies that the search key is a generic key whose length is specified in the KEYLENGTH option. The search for a record is satisfied when a record is found that has the same starting characters (generic key) as those specified.

GTEQ
(VSAM or data table) specifies that, if the search for a record that has the same key (complete or generic) as that specified in the RIDFLD option is unsuccessful, the first record that has a greater key satisfies the search.

This option is the default for directly browsing through a KSDS or an RRDS. It is not valid for directly browsing an ESDS, although it is valid for browsing through an ESDS using a path.

KEYLENGTH(*data-value***)**
specifies the length (halfword binary) of the key that has been specified in the RIDFLD option, except when RBA or RRN is specified, in which case KEYLENGTH is not valid.

This option must be specified if GENERIC is specified, and it can be specified whenever a key is specified. If the length specified is different from the length defined for the data set and the operation is not generic, the INVREQ condition occurs.

The INVREQ condition also occurs if a STARTBR command specifies GENERIC, and the KEYLENGTH value is not less than that specified in the VSAM definition.

If KEYLENGTH(0) is used with the object of positioning on the first record in the data set, the GTEQ option must also be specified. If EQUAL is specified either explicitly or by default with KEYLENGTH(0), the results of the STARTBR is unpredictable.

For remote files, the KEYLENGTH value can be specified in the FILE definition. If KEYLENGTH is not defined there, and is not specified in the application program, and the key is longer than 4 characters, the default value is 4.

RBA
(VSAM KSDS or ESDS base data sets, or CICS-maintained data tables only, not paths) specifies that the record identification field specified in the RIDFLD option contains a relative byte address. Use this option only when browsing an ESDS or KSDS base and using relative byte addresses instead of keys to identify the records.

You cannot use RBA for:

- User-maintained data tables
- Coupling facility data tables
- Any KSDS file opened in RLS access mode
- KSDS files that use extended addressing

Also, you are recommended not to use RBA for ESDS files that hold more than 4GB. (Use XRBA instead.)

REQID(*data-value***)**
specifies as a halfword binary value a unique request identifier for a browse, used to control multiple browse operations on the same or different data sets. If this option is not specified, a default value of zero is assumed.

RIDFLD(*data-area***)**
specifies the record identification field. The contents can be a key, a relative byte address, or relative record number (for VSAM data sets), or a block reference, physical key, and deblocking argument (for BDAM data sets). For a relative byte address or a relative record number, the format of this field must be fullword binary. For a relative byte address, the RIDFLD value can be greater than or equal to zero. For a relative record number, the RIDFLD value can be greater than or equal to 1.

See > and for more information about defining the record identification field.

For VSAM, a full record id of X'FF's indicates that the browse is to be positioned at the end of the data set in preparation for a backwards browse using READPREV commands.

RRN
(VSAM RRDS) specifies that the record identification field specified in the RIDFLD option contains a relative record number. This option should only be used with files referencing relative record data sets.

SYSID(*systemname***)**
specifies the name of the system to which the request is directed.

If you specify SYSID, and omit both RBA and RRN, you must also specify KEYLENGTH.

XRBA
specifies that the record identification field specified in the RIDFLD option contains an extended relative byte address. This option should be used when browsing records in an ESDS extended addressing data set.

If you specify XRBA on a STARTBR command, all other commands within the same browse must also specify XRBA.

KSDS data sets cannot be accessed by XRBA.

Conditions

84 DISABLED
RESP2 values:

50 A file is disabled. A file may be disabled because:
- It was initially defined as disabled and has not since been enabled.
- It has been disabled by a SET FILE or a CEMT SET FILE command.

Default action: terminate the task abnormally.

12 FILENOTFOUND
RESP2 values:

1 A file name referred to in the FILE option is not defined to CICS, and SYSID has not been specified.

Default action: terminate the task abnormally.

21 ILLOGIC
RESP2 values: (VSAM)

110 A VSAM error occurs that is not in one of the other CICS response categories.

See EIBRCODE in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

20 Browse operations are not allowed according to the resource definition.

25 The KEYLENGTH and GENERIC options are specified, and the length defined for the data set to which this file refers in the KEYLENGTH option is greater than or equal to the length of a full key.

26 The KEYLENGTH option is specified (but the GENERIC option is not specified), and the specified length does not equal the length defined for the data set to which this file refers.

33 An attempt is made to start a browse with a REQID already in use for another browse.

42 The KEYLENGTH and GENERIC options are specified, and the length specified in the KEYLENGTH option is less than zero.

44 The specified file is a user-maintained or coupling facility data table and the command does not conform to the format of STARTBR for such a data table (for example, RBA is specified).

51 A STARTBR command to a KSDS file that is being accessed in RLS mode specifies the RBA keyword. RLS mode does not support RBA access to KSDS files.

59 XRBA was specified, but the data set is not an extended ESDS.

Default action: terminate the task abnormally.

17 IOERR

RESP2 values:

120 There is an I/O error during the file control operation. An I/O error is any unusual event that is not covered by a CICS condition.

For VSAM files, IOERR usually indicates a hardware error. Further information is available in the EXEC interface block; for details, see EIB fields.

For a coupling facility data table, an IOERR indicates a bad response returned from a coupling facility access.

Default action: terminate the task abnormally.

54 ISCINVREQ

RESP2 values:

70 The remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

94 LOADING

RESP2 values:

104 The request cannot be satisfied because it is issued against a data table that is still being loaded. The condition can be raised for one of the following reasons:

- The STARTBR specifies a record that has not yet been loaded into a coupling facility data table. Records can be read while a CFDT is loading only if the requested key is within the range of those records already loaded.

  The LOADING response can also be returned for a coupling facility data table that has failed during loading. For more information about what happens if the load for a coupling facility data table fails, see the description of the XDTLC global user exit in Data tables management exits (XDTRD, XDTAD, and XDTLC).
- The READ specifies the GENERIC or GTEQ options for a user-maintained data table. While a UMT is being loaded, you can issue start browse requests with precise keys only.

If your application programs encounter the LOADING condition persistently or too frequently, check that this is not caused by conflicting file definitions that reference the same data set.

Default action: terminate the task abnormally.

70 NOTAUTH

RESP2 values:

101 A resource security check has failed on FILE(filename).

Default action: terminate the task abnormally.

13 NOTFND

RESP2 values:

80 An attempt to position on a record based on the search argument provided is unsuccessful.

81 XRBA was specified, and the value of RIDFLD was greater than 4 GB, but the data set is not an extended addressing ESDS.

NOTFND can also occur if a generic STARTBR with KEYLENGTH(0) specifies the EQUAL option.

Default action: terminate the task abnormally.

19 NOTOPEN

RESP2 values:

60 NOTOPEN (RESP2 60) is returned for one of the following reasons:

- The requested file is CLOSED and UNENABLED. The CLOSED, UNENABLED state is reached after a CLOSE request has been received against an OPEN ENABLED file and the file is no longer in use. You can also make CLOSED, UNENABLED the initial state, by specifying STATUS(UNENABLED) and OPENTIME(FIRSTREF) on the FILE resource definition. (For BDAM files, you use the FILSTAT parameter of the DFHFCT TYPE=FILE macro.)
- The requested file is OPEN and in use by other transactions, but a CLOSE request against the file has been received.
- A STARTBR command is issued against a data set that is quiesced, or is being quiesced, as a result of a SET DSNAME QUIESCED or IMMQUIESCED command.
- The requested file is CLOSED and ENABLED, so CICS has tried to open the file as part of executing the request. This file open has failed for some reason. You should examine the console for messages that explain why the file open has been unsuccessful.

This condition does not occur if the request is made to a CLOSED, DISABLED file. In this case, the DISABLED condition occurs.

Default action: terminate the task abnormally.

53 SYSIDERR

RESP2 values:

130 The SYSID option specifies a name that is neither the local system nor a remote system that is defined by a CONNECTION or IPCONN definition. SYSIDERR also occurs when the link to the remote system is known but unavailable. In the case of an IPCONN, SYSIDERR occurs if the link is known but either the local or remote systems do not support file control commands that are function shipped using IP interconnectivity.

131 For a coupling facility data table, the connection to the coupling facility data table server has failed. This could be because the server itself has failed, or the server is available, but CICS has failed to connect to it.

132 The start browse is operating on a coupling facility data table that no longer exists, probably because of a coupling facility failure, in which case the coupling facility data table server also fails. See Setting up and running a coupling facility data table server for information about restarting a coupling facility data table server and reloading a table.

Default action: terminate the task abnormally.

STARTBROWSE ACTIVITY

Start a browse of the child activities of a BTS activity, or of the descendant activities of a process.

STARTBROWSE ACTIVITY



Conditions: ACTIVITYERR, NOTAUTH, PROCESSERR

Description

STARTBROWSE ACTIVITY initializes a browse token which can be used to identify either:

- Each child activity of a specified BTS parent activity
- Each descendant activity of a specified BTS process.

If you specify the ACTIVITYID option, the children (but not the grandchildren nor other descendants) of the specified activity can be browsed. This option takes as its argument an activity identifier. This identifier may, for example, have been returned on a previous GETNEXT ACTIVITY command. If it was, the command starts a browse of child activities one level down the activity tree.

If you specify the PROCESS and PROCESSTYPE options, all the descendant activities of the specified process can be browsed. This type of browse is known as a **flat browse**. A flat browse is one which can return every descendant activity exactly once. A parent activity is always returned before its children. The value returned in the LEVEL option of a GETNEXT ACTIVITY command indicates the depth at which the activity lies in the process's activity-tree, with the root activity having a level of zero.

If you specify neither the ACTIVITYID nor the PROCESS and PROCESSTYPE options, the children of the current activity can be browsed.

Options

ACTIVITYID(data-value)
specifies the identifier (1–52 characters) of the activity whose child activities are to be browsed.

Typically, the activity identifier specified on this option has been returned on a previous GETNEXT ACTIVITY command (or, in the case of a root activity, on a GETNEXT PROCESS command). ACTIVITYID allows you to start a browse of child activities one level down the activity tree.

If you omit both this and the PROCESS option, the children of the current activity are browsed.

BROWSETOKEN(data-area)
specifies a fullword binary data area, into which CICS will place the browse token.

PROCESS(data-value)
specifies the name (1–36 characters) of the process whose descendant activities are to be browsed.

PROCESSTYPE(data-value)
specifies the process-type (1–8 characters) of the process named on the PROCESS option.

Conditions

109 ACTIVITYERR
RESP2 values:

1 The activity indicated by the ACTIVITYID option could not be found.

2 Because neither the ACTIVITYID nor the PROCESS options were specified, a browse of the children of the current activity was implied—but there is no current activity associated with the request.

19 The request timed out. It may be that another task using this activity-record has been prevented from ending.

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access the file whose data set contains the records to be browsed.

108 PROCESSERR
RESP2 values:

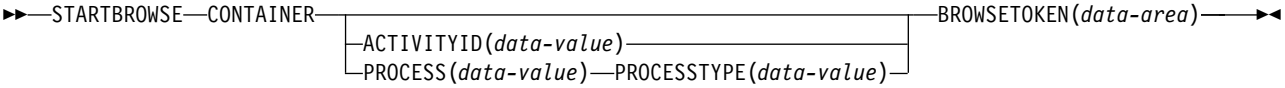
3 The process specified on the PROCESS option could not be found.

4 The process-type specified on the PROCESSTYPE option could not be found.

STARTBROWSE CONTAINER (BTS)

Start a browse of the containers associated with a BTS activity or process.

STARTBROWSE CONTAINER



Conditions: ACTIVITYERR, IOERR, NOTAUTH, PROCESSERR

This command is threadsafe.

Description

STARTBROWSE CONTAINER initializes a browse token which can be used to identify the name of each data-container associated with a specified BTS activity or process.

Note: The browse token should be used only by the program that issues the STARTBROWSE command.

If you do not specify the ACTIVITYI or the PROCESS option, CICS examines the context (channel or BTS) of the request. If a current channel exists, the command is assumed to be a command. If a current activity exists, its containers are browsed. If neither exists, an ACTIVITYERR 2 is raised: see the description of the ACTIVITYERR condition, below.

Options

ACTIVITYID(data-value)
specifies the identifier (1–52 characters) of the activity whose containers are to be browsed.

Typically, the identifier specified on this option has been returned on a previous GETNEXT ACTIVITY command.

BROWSETOKEN(data-area)
specifies a fullword binary data area, into which CICS will place the browse token.

PROCESS(data-value)
specifies the name (1–36 characters) of the process whose containers are to be browsed.

Note: The containers associated with a process (*process containers*) are globally available throughout the process. They are not the same as the root activity's containers.

PROCESSTYPE(data-value)
specifies the process-type (1–8 characters) of the process named on the PROCESS option.

Conditions

109 ACTIVITYERR
RESP2 values:

1 The activity indicated by the ACTIVITYID option could not be found.

2 None of the ACTIVITYID, PROCESS, or CHANNEL options were specified and there is no current channel and no current activity associated with the request.

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

17 IOERR
RESP2 values:

30 An input/output error has occurred on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access this resource in the way requested.

108 PROCESSERR
RESP2 values:

3 The process specified on the PROCESS option could not be found.

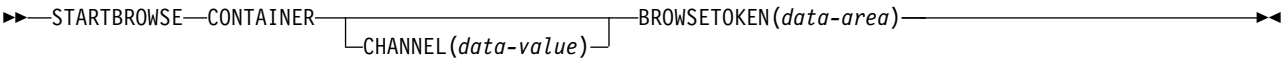
4 The process-type specified on the PROCESSTYPE option could not be found.

13 The request timed out. It may be that another task using this process-record has been prevented from ending.

STARTBROWSE CONTAINER (CHANNEL)

Start a browse of the containers associated with a channel.

STARTBROWSE CONTAINER



Conditions: CHANNELERR

This command is threadsafe.

Description

STARTBROWSE CONTAINER (CHANNEL) initializes a browse token which can be used to identify the name of each data-container associated with a specified channel.

Note: The browse token should be used only by the program that issues the STARTBROWSE command.

If you do not specify the CHANNEL option, CICS examines the channel of the request. If a current channel exists, its containers are browsed. If no current channel exists, the command is assumed to be a "STARTBROWSE CONTAINER (BTS)" on page 588 command.

Options

BROWSETOKEN(data-area)
specifies a fullword binary data area, into which CICS will place the browse token.

CHANNEL(data-value)
specifies the name (1–16 characters) of the channel whose containers are to be browsed. This must be the name of either the current channel or of a channel created by the program that issues the STARTBROWSE CONTAINER command. You can specify the channel name DFHTRANSACTION to use the transaction channel.

If this option is not specified, and the current context is channel, the current channel's containers are browsed.

The order in which containers are returned is undefined.

Conditions

122 CHANNELERR
RESP2 values:

2 The channel specified on the CHANNEL option could not be found.

STARTBROWSE EVENT

Start a browse of events known to a BTS activity.

STARTBROWSE EVENT



Conditions: ACTIVITYERR, INVREQ, IOERR, NOTAUTH

Description

STARTBROWSE EVENT initializes a browse token which can be used to identify each event (including each sub-event and system event) that is within the scope of a specified BTS activity. If you do not specify an activity, events within the scope of the current activity are browsed.

A browse started by STARTBROWSE EVENT returns:

- Atomic events. An atomic event returned on this command may or may not be included in the predicate of a composite event—that is, it may or may not be a sub-event.
- Composite events.
- System events.

Options

ACTIVITYID(data-value)
specifies the identifier (1–52 characters) of the activity whose events are to be browsed.

If you omit this option, events known to the current activity are browsed.

BROWSETOKEN(data-area)
specifies a fullword binary data area, into which CICS will place the browse token.

Conditions

109 ACTIVITYERR
RESP2 values:

1 The activity identifier specified on the ACTIVITYID option does not relate to any activity that is within the scope of this task.

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

16 INVREQ
RESP2 values:

1 There is no current activity within the scope of this task.

17 IOERR
RESP2 values:

30 An input/output error has occurred on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access this resource in the way requested.

STARTBROWSE PROCESS

Start a browse of all processes of a specified type within the CICS business transaction services system.

STARTBROWSE PROCESS



Conditions: IOERR, NOTAUTH, PROCESSERR

Description

STARTBROWSE PROCESS initializes a browse token which can be used to identify each process of a specified type within the CICS business transaction services system.

When you add a process to the BTS system, you use the PROCESSTYPE option of the DEFINE PROCESS command to categorize it. You specify the name of a PROCESSTYPE resource definition, which in turn names a CICS file definition that maps to a physical VSAM data set (the repository) on which details of the process and its constituent activities will be stored. (Records for multiple process-types can be stored on the same repository data set.)

The STARTBROWSE PROCESS command enables you to start a browse of processes of a specified type.

Options

BROWSETOKEN(data-area)
specifies a fullword binary data area, into which CICS will place the browse token.

PROCESSTYPE(data-value)
specifies the process-type (1–8 characters) of the processes to be browsed.

Conditions

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

70 NOTAUTH
RESP2 values:

101 The user associated with the issuing task is not authorized to access this resource in the way requested.

108 PROCESSERR
RESP2 values:

1 No processes of this process-type could be found.

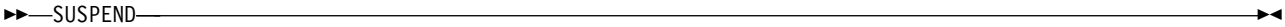
4 The process-type specified on the PROCESSTYPE option could not be found.

13 The request timed out. It may be that another task using this process-record has been prevented from ending.

SUSPEND

Suspend a task.

SUSPEND



This command is threadsafe.

Description

SUSPEND relinquishes control to a task of higher or equal dispatching priority. Control is returned to the task issuing the command as soon as no other task of a higher or equal priority is ready to be processed.

SUSPEND (BTS)

Suspend a BTS process or activity.

SUSPEND (BTS)



Conditions: ACTIVITYBUSY, ACTIVITYERR, INVREQ, IOERR, LOCKED, PROCESSERR

Description

SUSPEND (BTS) prevents a BTS process or activity being reattached when events in its event pool are fired.

The only process a program can suspend is the one that it has acquired in the current unit of work.

The only activities a program can suspend are as follows:

- If it is running as the activation of an activity, its own child activities. It can suspend several of its child activities within the same unit of work.
- The activity it has acquired, by means of an ACQUIRE ACTIVITYID command, in the current unit of work.

To resume a suspended process or activity, a RESUME command must be issued.

Options

ACQACTIVITY
specifies that the activity to be suspended is the one that the current unit of work has acquired by means of an ACQUIRE ACTIVITYID command.

ACQPROCESS
specifies that the process that is currently acquired by the requestor is to be suspended.

ACTIVITY(data-value)
specifies the name (1–16 characters) of the child activity to be suspended.

Conditions

107 ACTIVITYBUSY
RESP2 values:

19 The request timed out. It may be that another task using this activity-record has been prevented from ending.

109 ACTIVITYERR
RESP2 values:

8 The activity named on the ACTIVITY option could not be found.

16 INVREQ
RESP2 values:

4 The ACTIVITY option was used to name a child activity, but the command was issued outside the scope of a currently-active activity.

14 The activity is in COMPLETE or CANCELLING mode, and therefore cannot be suspended.

15 The ACQPROCESS option was used, but the unit of work that issued the request has not acquired a process.

24 The ACQACTIVITY option was used, but the unit of work that issued the request has not acquired an activity.

17 IOERR
RESP2 values:

29 The repository file is unavailable.

30 An input/output error has occurred on the repository file.

100 LOCKED
The request cannot be performed because a retained lock exists against the relevant record on the repository file.

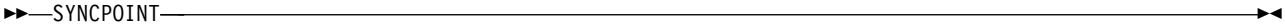
108 PROCESSERR
RESP2 values:

5 The process could not be found.

SYNCPOINT

Establish a syncpoint.

SYNCPOINT



Conditions: INVREQ, ROLLEDBACK

This command is threadsafe.

Note: The Recovery Manager processes this command on an open TCB wherever possible to minimize TCB switching. Syncpoint processing can take place on an open TCB for all resource types declared as threadsafe that were accessed in the unit of work. If resource types not declared as threadsafe were accessed in the unit of work, the Recovery Manager switches to the QR TCB for those resource types. A CICS resource type declares itself to the Recovery Manager as threadsafe if the EXEC CICS commands relating to the resource type are threadsafe.

Description

SYNCPOINT divides a task (usually a long-running one) into smaller units of work. It specifies that all changes to recoverable resources made by the task since its last syncpoint are to be committed.

Note: A failure occurring during the commit phase (phase 2) of syncpoint processing does not return an error condition and the transaction is not abnormally terminated. Subsequent units of work in the transaction are allowed to continue normally. See the Unit of work recovery and abend processing for further information.

Conditions

16 INVREQ
RESP2 values:

200 SYNCPOINT was in a program that is linked to from a remote system that has not specified the SYNCONRETURN option, or if it has been linked to locally and is defined with EXECUTIONSET=DPLSUBSET.

Default action: terminate the task abnormally.

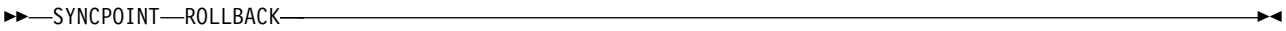
82 ROLLEDBACK
occurs when a SYNCPOINT command is driven into rollback by a remote system that is unable to commit the syncpoint. All changes made to recoverable resources in the current unit of work are backed out.

Default action: terminate the task abnormally.

SYNCPOINT ROLLBACK

Back out to last syncpoint.

SYNCPOINT ROLLBACK



Condition: INVREQ

This command is threadsafe.

Note: The Recovery Manager processes this command on an open TCB wherever possible to minimize TCB switching. Syncpoint processing can take place on an open TCB for all resource types declared as threadsafe that were accessed in the unit of work. If resource types not declared as threadsafe were accessed in the unit of work, the Recovery Manager switches to the QR TCB for those resource types. A CICS resource type declares itself to the Recovery Manager as threadsafe if the EXEC CICS commands relating to the resource type are threadsafe.

Options

ROLLBACK
specifies that all changes to recoverable resources made by the task since its last syncpoint are to be backed out.

This option can be used, for example, to tidy up in a HANDLE ABEND routine, or to revoke database changes after the application program finds irrecoverable errors in its input data.

If the unit of work updates remote recoverable resources using an MRO or APPC session, the ROLLBACK option is propagated to the back-end transaction.

When a distributed transaction processing conversation is in use, the remote application program has the EIB fields EIBSYNRB, EIBERR, and EIBERRCD set. For the conversation to continue, the remote application program should execute a SYNCPOINT ROLLBACK command.

When the mirror transaction is involved in the unit of work using an MRO or APPC session, the mirror honors the rollback request, revokes changes, and then terminates normally.

This option is not supported across LUTYPE6.1 z/OS Communications Server sessions to the mirror or back-end transactions. In these cases, the front-end transactions could be abended to cause the back-end transactions to back out.

Note: A failure occurring during the backout phase (phase 2) of syncpoint processing does not return an error condition and the transaction is not abnormally terminated. Subsequent units of work in the transaction are allowed to continue normally. See Unit of work recovery and abend processing for further information.

Note: A deferred EXEC CICS SEND request is canceled during a **SYNCPOINT ROLLBACK** command.

Conditions

16 INVREQ
RESP2 values:

200 SYNCPOINT ROLLBACK was in a program that is linked to from a remote system that has not specified the SYNCONRETURN option, or if it has been linked to locally and is defined with EXECUTIONSET=DPLSUBSET

Default action: terminate the task abnormally.

TEST EVENT

Test whether a BTS event has fired.

TEST EVENT



Conditions: EVENTERR, INVREQ

Description

TEST EVENT tests whether a named BTS event has occurred (fired).

Options

EVENT(data-value)
specifies the name (1–16 characters) of the event to test for completion.

FIRESTATUS(cvda)
FIRESTATUS returns the fire status of the event. CVDA values are:

FIRED
The event has fired.

NOTFIRED
The event has not fired.

Conditions

111 EVENTERR
RESP2 values:

4 The event specified on the EVENT option is not recognized by BTS.

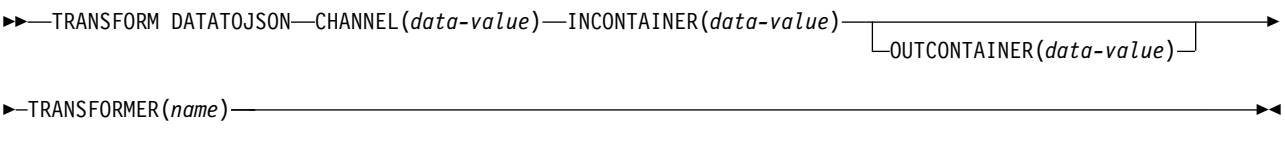
16 INVREQ
RESP2 values:

1 The command was issued outside the scope of an activity.

TRANSFORM DATATOJSON

Use the **TRANSFORM DATATOJSON** command to convert application data to JSON.

TRANSFORM DATATOJSON



Conditions: CHANNELERR, CONTAINERERR, INVREQ, NOTFND

This command is threadsafe.

Description

The **TRANSFORM DATATOJSON** command transforms application data to JSON using mappings that are defined in the JSON binding. The JSON transformer resource is a bundle that defines where the JSON binding and the JSON schema are located.

Options

CHANNEL(*data-value*)
Specify the name of the channel used to pass the containers that hold the OUTCONTAINER and INCONTAINER data. The name of the channel must be 16 characters in length. If the channel name is less than 16 characters, you must pad the data value with trailing blanks. You can specify DFHTRANSACTION to use the transaction channel.

INCONTAINER(*data-value*)
Specify the name of the container that contains the application data to be converted. This container must be present on the channel. The name of the container must be 16 characters in length. If the container name is less than 16 characters, you must pad the data value with trailing blanks.

CICS reads from this container in BIT mode.

OUTCONTAINER(*data-value*)

Optionally specify the name of the container to contain the JSON output when the command completes. The name of the container must be 16 characters in length. If the container name is less than 16 characters, you must pad the data value with trailing blanks. If this option is omitted, DFHJSON-JSON is used.

Do not create the target container before you issue the command because one container is created and populated as part of the command itself. If the container does exist, it is deleted before the transformation takes place. CICS populates this container in CHAR mode.

TRANSFORMER(*name*)
Specify the 16-byte name of the JSON transformer resource bundle that CICS uses to transform the data to JSON. The resource defines the JSON binding and the JSON schema. The name of the JSON transformer resource must be 16 characters in length. If the resource name is less than 16 characters, you must pad the value with blanks.

This resource bundle must be installed and enabled for this command to work.

Conditions

122 CHANNELERR
RESP2 values:

1 The channel specified by the **CHANNEL** parameter is incorrect.

2 The channel specified by the **CHANNEL** parameter was not found.

110 CONTAINERERR
RESP2 values:

3 The container specified by the **INCONTAINER** parameter was not found.

16 INVREQ
RESP2 values:

1 The JSON transformer bundle resource is not enabled.

6 The application data cannot be converted. For more information, see the error message in the DFH-JSON-ERRORMSG container.

8 The INCONTAINER container is not populated in BIT mode.

101 The user is not authorized to use the JSON transformer.

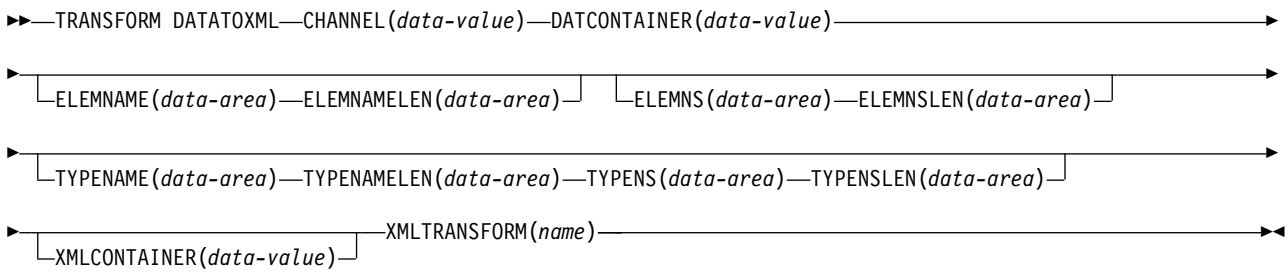
13 NOTFND
RESP2 values:

1 The JSON transformer bundle resource was not found.

TRANSFORM DATATOXML

Use the **TRANSFORM DATATOXML** command to convert application data to XML.

TRANSFORM DATATOXML



Conditions: NOTFND, LENGERR, CHANNELERR, CONTAINERERR, INVREQ

This command is threadsafe.

Description

The **TRANSFORM DATATOXML** command transforms application data to XML using mappings that are defined in the XML binding. The XMLTRANSFORM resource defines where the XML binding and the XML schema are located.

Options

CHANNEL(*data-value*)
Specify the name of the channel used to pass the containers holding the XMLCONTAINER and DATCONTAINER data. The name of the channel must be 16 characters in length. If the channel name is less than 16 characters, you must pad the data value with trailing blanks. You can specify the channel name DFHTRANSACTION to use the transaction channel.

DATCONTAINER(*data-value*)
Specify the name of the container that contains the application data for conversion. This container must be present on the channel. The name of the container must be 16 characters in length. If the container name is less than 16 characters, you must pad the data value with trailing blanks.

CICS reads from this container in BIT mode.

ELEMNAME(*data-area*)
Specify the name of an XML element. CICS returns the local name of the XML element that CICS generates.

ELEMNAMELEN(*data-area*)
Specify the fullword binary length of the XML element in the ELEMNAME option. The maximum value of ELEMSNAMELEN is 255.

ELEMNS(*data-area*)
Specify the namespace URI of the XML element that is returned in the ELEMNAME option.

ELEMNSLEN(*data-area*)
Specify the fullword binary length of the namespace in the ELEMNS option. The maximum value of ELEMNSLEN is 255.

TYPENAME(*data-area*)
Specify the xsi:type of the XML element that is returned in the ELEMNAME option.

TYPENAMELEN(*data-area*)
Specify the fullword binary length of the xsi:type that is returned in the TYPENAME option.

TYPENS(*data-area*)
Specify the namespace of the xsi:type attribute of the XML element that is returned in the ELEMNAME option.

TYPENSLEN(*data-area*)
Specify the fullword binary length of the namespace for the xsi:type attribute that is returned in the TYPENS option.

XMLCONTAINER(*data-value*)
Specify the name of the container that will contain the output XML once the command completes. The name of the container must be 16 characters in length. If the container name is less than 16 characters, you must pad the data value with trailing blanks.

You do not need to create the target container before issuing the command. The container is created and populated as part of the command itself. If the container does exist, and has been defined as a data type of BIT, it is deleted and redefined as type CHAR as part of the command.

XMLTRANSFORM(*data-value*)
Specify the name of the XMLTRANSFORM resource that CICS uses to transform the data to XML. The resource defines the XML binding and the XML schema. The name of the XMLTRANSFORM resource must be 32 characters. If the resource name is less than 32 characters, you must pad the value with blanks.

Conditions

13 NOTFND
RESP2 values:

1 The XMLTRANSFORM was not found.

122 CHANNELERR
RESP2 values:

1 The channel specified by the **CHANNEL** parameter is incorrect.

2 The channel specified by the **CHANNEL** parameter was not found.

110 CONTAINERERR
RESP2 values:

1 The container specified by the **XMLCONTAINER** parameter was not found.

2 The container specified by the **NSCONTAINER** parameter was not found.

3 The container specified by the **DATCONTAINER** parameter was not found.

22 LENGERR
RESP2 values:

1 The data in the container specified by the **DATCONTAINER** parameter is too short for the specified transformation type.

2 The **ELEMNAME** buffer is too small.

3 The **ELEMNS** buffer is too small.

4 The **TYPENAME** buffer is too small.

5 The **TYPENS** buffer is too small.

6 The **ELEMNAMELEN** value exceeds the maximum value of 255.

7 The **ELEMNSLEN** value exceeds the maximum value of 255.

16 INVREQ
RESP2 values:

1 The XMLTRANSFORM resource is not enabled.

2 The **XMLCONTAINER** container is empty.

3 The XML input data is invalid. For more information, see the error message in the DFH-XML-ERRORMSG container.

4 The XML input data cannot be converted. For more information, see the error message in the DFH-XML-ERRORMSG container.

5 The application data is invalid. For more information, see the error message in the DFH-XML-ERRORMSG container.

6 The application data cannot be converted. For more information, see the error message in the DFH-XML-ERRORMSG container.

8 The application data container is not populated in BIT mode

9 The XMLTRANSFORM does not support the requested XML element.

10 The XMLTRANSFORM does not support the requested XML type.

11 There was a problem linking to a vendor-supplied transformer program.

13 The **CHANNEL** parameter was not supplied and is required.

14 The **ELEMNAME** parameter was not supplied and is required.

15 The **ELEMNS** parameter was not supplied and is required.

16 The **DATCONTAINER** parameter was not set and is required.

17 There is a runtime validation failure.

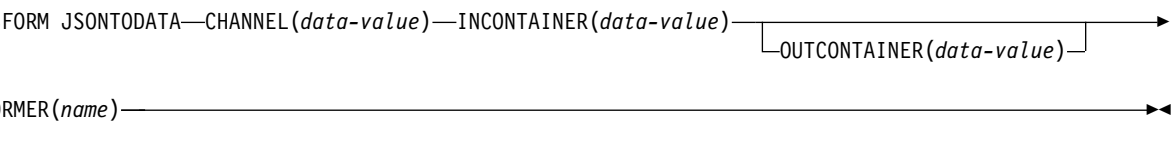
18 There is a container datatype error.

101 The user is not authorized to use the XMLTRANSFORM.

TRANSFORM JSONTODATA

Use the **TRANSFORM JSONTODATA** command to convert JSON to application data.

TRANSFORM JSONTODATA



Conditions: CHANNELERR, CONTAINERERR, INVREQ, NOTFND

This command is threadsafe.

Description

The **TRANSFORM JSONTODATA** command transforms JSON to application data. The JSON transformer resource is a bundle that defines the location of the JSON binding and the JSON schema for transforming JSON to application data.

Options

CHANNEL(*data-value*)
Specify the name of the channel used to pass the containers that hold the OUTCONTAINER and INCONTAINER data. The name of the channel must be 16 characters in length. If the channel name is less than 16 characters, you must pad the data value with trailing blanks. You can specify DFHTRANSACTION to use the transaction channel.

INCONTAINER(*data-value*)
Specify the 16-byte name of the container that contains the JSON to be converted. This container must already exist and be populated in CHAR mode. If it is populated in BIT mode, CICS tries to determine the data encoding.

OUTCONTAINER(*data-value*)

Optionally specify the name of the output container to contain the converted data. The name of the container must be 16 characters in length. If the container name is less than 16 characters, you must pad the data value with trailing blanks. If this option is omitted, DFHJSON-DATA is used.

Do not create the target container before you issue the command because one container is created and populated as part of the command itself. If the container does exist, it is deleted before the transformation takes place. CICS populates this container in BIT mode.

TRANSFORMER(*name*)

Specify the 16-byte name of the JSON transformer resource bundle that CICS uses to transform the data to JSON. The resource defines the JSON binding

and the JSON schema. The name of the JSON transformer resource must be 16 characters in length. If the resource name is less than 16 characters, you must pad the value with blanks.

Conditions

122 CHANNELERR
RESP2 values:

1 The channel specified by the **CHANNEL** parameter is incorrect.

2 The channel specified by the **CHANNEL** parameter was not found.

110 CONTAINERERR
RESP2 values:

1 The container specified by the **INCONTAINER** parameter was not found.

16 INVREQ
RESP2 values:

1 The JSON transformer bundle resource is not enabled.

4 The application data cannot be converted. For more information, see the error message in the DFH-JSON-ERRORMSG container.

7 The INCONTAINER container is not populated in CHAR mode.

101 The user is not authorized to use the JSON transformer.

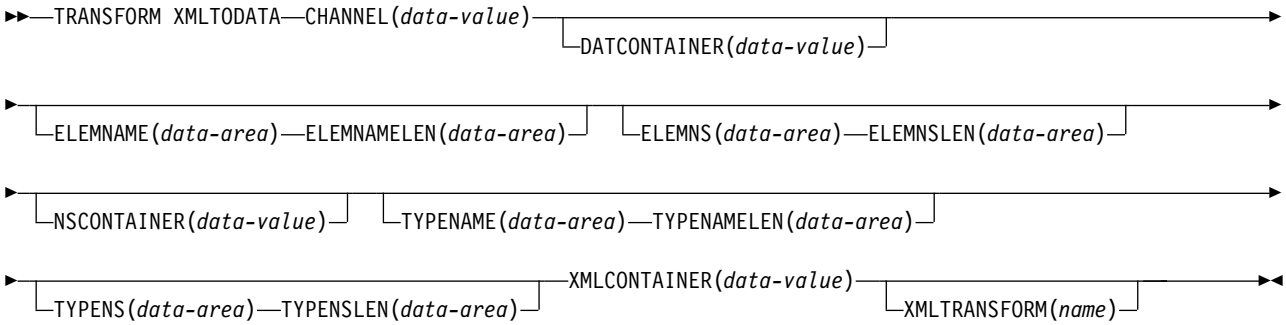
13 NOTFND
RESP2 values:

1 The JSON transformer bundle resource was not found.

TRANSFORM XMLTODATA

Use the **TRANSFORM XMLTODATA** command to convert XML to application data.

TRANSFORM XMLTODATA



Conditions: CHANNELERR, CONTAINERERR, INVREQ, LENGERR, NOTFND,

This command is threadsafe.

Description

The **TRANSFORM XMLTODATA** command can either transform XML to application data or query XML to return information about the XML elements to the application

program. The XMLTRANSFORM resource defines the location of the XML binding and schema for transforming the XML to application data. If you do not specify an XMLTRANSFORM resource on the command, CICS queries the XML instead.

Options

CHANNEL(*data-value*)
Specify the 16-byte name of the channel used to pass the containers holding the XMLCONTAINER and DATCONTAINER data. The name of the channel must be 16 characters in length. If the channel name is less than 16 characters, you must pad the data value with trailing blanks. You can specify the channel name `DFHTRANSACTION` to use the transaction channel.

DATCONTAINER(*data-value*)
Specify the 16-byte name of the output container that CICS populates with the converted data. The name of the container must be 16 characters in length. If the container name is less than 16 characters, you must pad the data value with trailing blanks.

CICS populates this container in BIT mode.

ELEMNAME(*data-area*)
Specify an input value to return the name of an XML element. CICS populates the ELEMNAME option with the local name of the first XML element that it finds in the XMLCONTAINER container. The application must also specify an input value for the ELEMNAMELEN option that indicates the maximum length of the data area.

ELEMNAMELEN(*data-area*)
Specify an input value to return the fullword binary length of the XML element in the ELEMNAME option. CICS updates the value of the ELEMNAMELEN option to indicate the real length of the element name that it finds.

ELEMNS(*data-area*)
Specify an input value to return the namespace URI of the XML element to which the ELEMNAME option refers. CICS populates this parameter with the namespace URI of the first XML element that it finds in the XMLCONTAINER container. The application must also specify an input value for the ELEMNSLEN option that indicates the maximum length of the data area.

ELEMNSLEN(*data-area*)
Specify an input value to return the length of the ELEMNS option. CICS updates the value of the ELEMNSLEN option to indicate the real length of the namespace URI.

NSCONTAINER(*data-value*)

Specify the 16-byte name of the container that contains a list of XML namespace declarations that are in scope. These XML namespace declarations can be referenced in the body of the XMLCONTAINER container. The container must be populated in CHAR mode.

TYPENAME(*data-area*)
Specify an input value to return the `xsi:type` of the XML element referred to by the ELEMNAME option. This parameter is populated by CICS with the local name of the `xsi:type` attribute of the first XML tag that is found in the XMLCONTAINER container. If the first XML tag does not have an `xsi:type` attribute, this parameter remains empty. If the application supplies a value for the TYPENAME option, this value overrides any element and type information

in the supplied XML and CICS attempts the transformation using the type name supplied by this application (together with the associated TYPENS option).

TYPENAMELEN(*data-area*)
Specify an input value to return the length of the TYPENAME option.

TYPENS(*data-area*)
Specify an input value to return the namespace of the `xsi:type` attribute of the XML element to which the ELEMNAME option refers. This parameter is populated by CICS with the namespace of the `xsi:type` attribute of the first XML element that is found in the XMLCONTAINER container.

TYPENSLEN(*data-area*)
Specify an input value to return the length of the TYPENS option.

XMLCONTAINER(*data-value*)
Specify the 16-byte name of the input container that contains the XML to be converted. This container must already exist and be populated in CHAR mode. If it is populated in BIT mode, CICS tries to determine the data encoding.

XMLTRANSFORM(*name*)
Specify the 32-byte name of the XMLTRANSFORM resource that CICS uses to transform the data to XML. The resource defines the XML binding and the XML schema. The name of the XMLTRANSFORM resource must be 32 characters. If the resource name is less than 32 characters, you must pad the value with blanks.

If you do not specify the XMLTRANSFORM option, no data transformation occurs. Instead, the application queries the XML. CICS returns the XML element and type information in the ELEMNAME, ELEMNS, TYPENAME, and TYPENS options.

Conditions

13 NOTFND
RESP2 values:

1 The XMLTRANSFORM was not found.

122 CHANNELERR
RESP2 values:

1 The channel specified by the **CHANNEL** parameter is incorrect.

2 The channel specified by the **CHANNEL** parameter was not found.

110 CONTAINERERR
RESP2 values:

1 The container specified by the **XMLCONTAINER** parameter was not found.

2 The container specified by the **NSCONTAINER** parameter was not found.

3 The container specified by the **DATCONTAINER** parameter was not found.

22 LENGERR
RESP2 values:

1 The data in the container specified by the **DATCONTAINER** parameter is too short for the specified transformation type.

2 The **ELEMNAME** buffer is too small.

3 The **ELEMNS** buffer is too small.

4 The **TYPENAME** buffer is too small.

5 The **TYPENS** buffer is too small.

16 INVREQ
RESP2 values:

1 The XMLTRANSFORM resource is not enabled.

2 The **XMLCONTAINER** container is empty.

3 The XML input data is incorrect. For more information, see the error message in the DFH-XML-ERRORMSG container.

4 The XML input data cannot be converted. For more information, see the error message in the DFH-XML-ERRORMSG container.

5 The application data is incorrect. For more information, see the error message in the DFH-XML-ERRORMSG container.

6 The application data cannot be converted. For more information, see the error message in the DFH-XML-ERRORMSG container.

7 Either the XML container or the NAMESPACE container is not CHAR mode.

9 The XMLTRANSFORM does not support the requested XML element.

10 The XMLTRANSFORM does not support the requested XML type.

11 There was a problem linking to a vendor-supplied transformer program.

13 The **CHANNEL** parameter was not supplied and is required.

14 The **ELEMNAME** parameter was not supplied and is required.

15 The **ELEMNS** parameter was not supplied and is required.

16 The **DATCONTAINER** parameter was not set and is required.

17 There is a runtime validation failure.

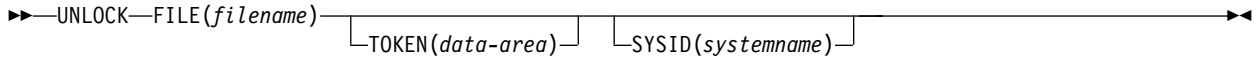
18 There is a container datatype error.

101 The user is not authorized to use the XMLTRANSFORM.

UNLOCK

Release exclusive control.

UNLOCK



Conditions: DISABLED, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, NOTAUTH, NOTOPEN, SYSIDERR

This command is threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over an IPIC connection to a remote CICS region.
- Defined as either local VSAM or RLS.

This command is not threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over a non-IPIC connection.
- Defined as a shared data table, coupling facility data table, or BDAM file.

Description

UNLOCK releases the exclusive control established in response to a read command with the UPDATE option. You use it if you retrieve a record for update, and then decide that you do not want to update the record after all. However, for a recoverable file (other than one that refers to a coupling facility data table), the resource remains locked until either a SYNCPOINT command is executed or the task is terminated. The record can be in a data set, or in a CICS or user-maintained data table, on a local or a remote system.

If the UNLOCK command refers to a record in a recoverable coupling facility data table the record lock is released immediately provided that the task has not made any previous change to the same record (or added it as a new record) within the current unit of work. If changes have been made to the record, or it is a new record added to the table, it remains locked until either a SYNCPOINT command is executed or the task is terminated.

If an UNLOCK command does not have a token, an attempt is made to match it to either a read with the UPDATE option that also does not have a token, or to a WRITE MASSINSERT operation. If neither of these is found, no action is taken and a NORMAL response is returned.

Use this command to terminate a VSAM WRITE MASSINSERT operation against a VSAM file.

Releasing locks when updating in browse

The UNLOCK command does not release locks held against records locked in response to READNEXT or READPREV commands that specify the update option. It only invalidates the TOKEN value so that it cannot be used to complete an update.

Options

FILE(*filename***)**
specifies the name of the file to be released.

If SYSID is specified, the data set to which this file refers is assumed to be on a remote system irrespective of whether the name is defined to CICS. Otherwise, the resource definition is used to find out whether the data set is on a local or a remote system.

SYSID(*systemname***)**
specifies the name of the system to which the request is directed.

TOKEN(*data-area***)**
specifies as a fullword binary value a unique request identifier for an UNLOCK, used to associate it with a previous READ, READNEXT, or READPREV command that specified the UPDATE option.

If you specify UNLOCK with the TOKEN returned on a READNEXT UPDATE or READPREV UPDATE command for a file accessed in RLS mode, the UNLOCK command invalidates the TOKEN value so that it cannot be used to complete an update. It does not release the record lock.

TOKEN can be function shipped. However, if a request specifying TOKEN is function shipped to a CICS region that does not support this keyword, the request fails.

Conditions

84 DISABLED
RESP2 values:

50 A file is disabled because it was initially defined as disabled and has not since been enabled.

A file is disabled by an EXEC CICS SET FILE or a CEMT SET FILE command.

This condition cannot occur when the UNLOCK follows a successful read for update or a VSAM WRITE MASSINSERT.

Default action: terminate the task abnormally.

12 FILENOTFOUND
RESP2 values:

1 A file name referred to in the FILE option is not defined to CICS and SYSID has not been specified.

Default action: terminate the task abnormally.

21 ILLOGIC
RESP2 values: (VSAM and CICS-maintained data tables)

110 A VSAM error occurs that is not in one of the other CICS response categories.

See EIBRCODE in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

47 An unlock includes a token whose value cannot be matched against any token in use for an existing READ with the UPDATE option.

48 An attempt is made to function-ship a request that includes a TOKEN keyword.

Default action: terminate the task abnormally.

17 IOERR

RESP2 values:

120 There is an I/O error during the file control operation. An I/O error is any unusual event that is not covered by a CICS condition.

For VSAM files, IOERR usually indicates a hardware error. Further information is available in the EXEC interface block; for details, see EIB fields.

For a coupling facility data table, an IOERR indicates a bad response returned from a coupling facility access.

Default action: terminate the task abnormally.

54 ISCINVREQ

RESP2 values:

70 The remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

70 NOTAUTH

RESP2 values:

101 A resource security check has failed on FILE(filename).

Default action: terminate the task abnormally.

19 NOTOPEN

RESP2 values:

60 NOTOPEN (RESP2 60) is returned for one of the following reasons:

- The requested file is CLOSED and UNENABLED. The CLOSED, UNENABLED state is reached after a CLOSE request has been received against an OPEN ENABLED file and the file is no longer in use. You can also make CLOSED, UNENABLED the initial state, by specifying STATUS(UNENABLED) and OPENTIME(FIRSTREF) on the FILE resource definition. (For BDAM files, you use the FILSTAT parameter of the DFHFCT TYPE=FILE macro.)
- The requested file is OPEN and in use by other transactions, but a CLOSE request against the file has been received.
- An UNLOCK command is issued against a data set that is quiesced, or is being quiesced, as a result of a SET DSNAME QUIESCED or IMMQUIESCED command.
- The requested file is CLOSED and ENABLED, so CICS has tried to open the file as part of executing the request. This file open has failed for some reason. You should examine the console for messages that explain why the file open has been unsuccessful.

This condition does not occur if the request is made to a CLOSED, DISABLED file. In this case, the DISABLED condition occurs.

It also cannot occur when the UNLOCK follows a successful READ for update or a WRITE MASSINSERT operation.

Default action: terminate the task abnormally.

53 SYSIDERR

RESP2 values:

130 The SYSID option specifies a name that is neither the local CICS region nor a remote system defined to CICS by a CONNECTION definition. SYSIDERR also occurs when the link to the remote system is closed.

131 For a coupling facility data table, the connection to the coupling facility data table server has failed. This could be because the server itself has failed, or the server is available, but CICS has failed to connect to it.

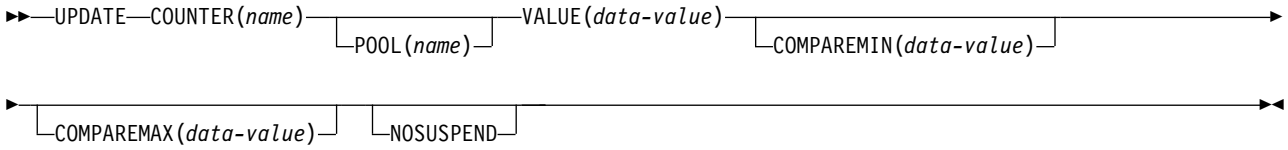
132 The UNLOCK is issued against a coupling facility data table that no longer exists, probably because of a coupling facility failure, in which case the coupling facility data table server also fails. See the Defining and starting a coupling facility data table server region for information about restarting a coupling facility data table server and reloading a table.

Default action: terminate the task abnormally.

UPDATE COUNTER and UPDATE DCOUNTER

Update the current value of a named counter. Use COUNTER for fullword signed counters and DCOUNTER for doubleword unsigned counters.

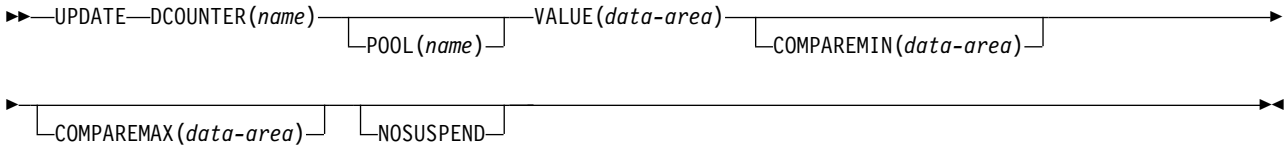
UPDATE COUNTER



Conditions: BUSY, INVREQ, SUPPRESSED

This command is threadsafe.

UPDATE DCOUNTER



Conditions: BUSY, INVREQ, SUPPRESSED

This command is threadsafe.

Description

These counter commands set a new current value for the named counter.

You can use the COMPAREMAX and COMPAREMIN options to set a new current value only if it falls within a specified range, or is above or below a specified value.

For information about specifying fullword and doubleword variables on these named counter commands, see "CICS command argument values" on page 3.

Options

COMPAREMAX(*data-value***)**
Specifies, as a fullword signed binary value (or doubleword unsigned binary value for DCOUNTER), a value to compare with the current value of the named counter, and makes the result of the UPDATE command conditional on the comparison:

- If the current value to assign is less than, or equal to, the value specified for the COMPAREMAX parameter, the current value is reset and the response is normal.
- If the current value is greater than the specified value, CICS returns an exception condition.

Normally, the COMPAREMAX value is greater than the COMPAREMIN value and the current value must satisfy both comparisons (that is, it must be between the two values or equal to one of them).

You can specify a COMPAREMAX value that is less than the COMPAREMIN value. In this situation, the current value is considered to be in range if it satisfies either the COMPAREMIN or the COMPAREMAX comparison.

COMPAREMIN(*data-value***)**
Specifies, as a fullword signed binary value (or doubleword unsigned binary value for DCOUNTER), a value to compare with the current value of the named counter, and makes the result of the UPDATE command conditional on the comparison:

- If the current value to assign is equal to, or greater than, the value specified for the COMPAREMIN parameter, the current value is reset and the response is normal.
- If the current value is less than the specified value, CICS returns an exception condition.

Note: You can specify a COMPAREMIN value that is greater than the COMPAREMAX value. See the COMPAREMAX parameter for the effect of this.

COUNTER(*name***)**
Specifies the name of the named counter for which the current number is to be reset to the value specified on the value parameter. The name can be up to 16 alphanumeric characters. If *name* is a variable that contains a name that is less than 16 characters, the name must be padded with trailing blanks.

DCOUNTER(*name***)**
Specifies the name of the named counter for which the current number is to be reset to the value specified on the value parameter. The name can be up to 16 alphanumeric characters. If *name* is a variable that contains a name that is less than 16 characters, the name must be padded with trailing blanks.

NOSUSPEND
Specifies an immediate return to the application program during a coupling facility structure rebuild. If you omit the NOSUSPEND parameter, the request is retried until it is successful. If you specify NOSUSPEND, the request fails immediately, and CICS returns BUSY, RESP2=500.

POOL(*poolname***)**
Specifies the name of the pool in which the named counter resides.

Valid characters for the pool selector string are A through Z, 0 through 9, $ @ # and _ (underscore). If *name* is a variable that contains a name that is less than 8 characters, the name must be padded with trailing blanks.

This parameter is optional. If you omit the name of the pool, a pool selector value of 8 blanks is assumed.

If there is no matching entry in the DFHNCOPT options table, CICS uses the name specified on the NCPLDFT system initialization parameter, which specifies the default named counter pool.

For information about generating a named counter options table using the DFHNCO macro, see .

VALUE(*data-value***)**
Specifies the new number to set as the current value for the named counter, using a fullword signed binary value for COUNTER and a doubleword unsigned value for DCOUNTER.

Conditions

128 BUSY
RESP2 values:

500 The NOSUSPEND option was specified on the command, and the coupling facility structure is not currently available during a rebuild.

Default action: terminate the task abnormally.

16 INVREQ
RESP2 values:

201 Named counter not found.

301 The server has reported an error code that is not understood by the named counter interface. Generally, this is not possible unless the interface load module, DFHNCIF, is at a lower maintenance or release level than the server itself.

303 An unexpected error, such as structure failure or loss of connectivity, has occurred on a macro used to access the coupling facility. Further information is in message DFHNC0441 in the application job log.

304 The pool selection parameter specified in the program cannot be resolved to a valid server name using the current options table.

305 The interface cannot establish a connection to the server for the selected named counter pool. Further information is in an AXM services message (AXMSC*nnnn*) in the application job log.

306 An abend occurred during server processing of a request. Further information is in a message in the application job log and the server job log.

308 The DFHNCOPT options table module, required to resolve a pool name, cannot be loaded.

309 During processing of the options table, the named counter interface encountered an unknown entry format. Either the options table is not correctly generated, or the DFHNCIF interface load module is not at the same release level as the options table.

310 An options table entry that matches the given pool name specified a user exit program, but the user exit program is not link-edited with the options table and cannot be loaded.

311 A response from the named counter server to the client region interface module, DFHNCIF, indicates that a system-managed rebuild is in progress but the EXEC CICS interface does not recognize the condition. This means that the CICS region is at CICS TS 2.1 or earlier.

403 The POOL parameter contains invalid characters or embedded spaces.

404 The COUNTER parameter contains invalid characters or embedded spaces.

406 The VALUE parameter is invalid. You cannot set the current value to less than the minimum value, or greater than the maximum value plus 1.

Default action: terminate the task abnormally.

72 SUPPRESSED

RESP2 values:

103 One of the following:

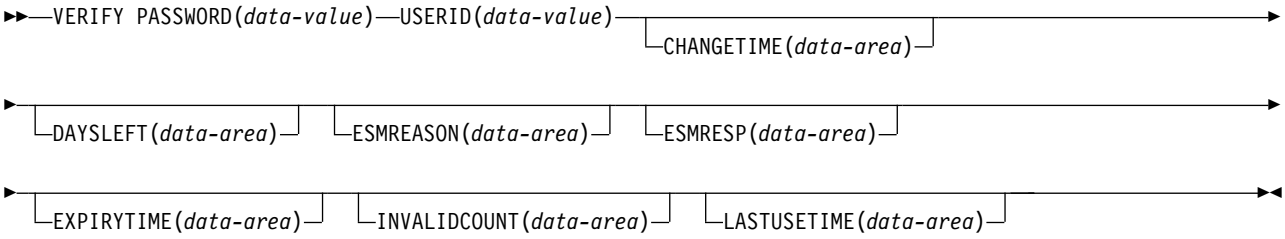
- The current value of the named counter is not within the range specified by the COMPAREMAX and COMPAREMIN parameters, when both are specified
- The current value of the named counter is greater than the COMPAREMAX parameter or less than the COMPAREMIN parameter, when only one option is specified.

Default action: terminate the task abnormally.

VERIFY PASSWORD

Verify that a password matches the password recorded by an external security manager.

VERIFY PASSWORD



Conditions: INVREQ, NOTAUTH, USERIDERR

This command is threadsafe.

Description

Use the **VERIFY PASSWORD** command to check that a password matches the password recorded by an external security manager for a user ID. The command returns the values recorded by the external security manager for the password. This process is called password verification. If your system uses password phrases in addition to or instead of standard passwords, use the **VERIFY PHRASE** command instead of the **VERIFY PASSWORD** command.

Attention: To ensure that passwords are not revealed in system or transaction dumps, clear the password or password phrase fields on the EXEC CICS commands that have a password or password phrase option as soon as possible after use.

If a **VERIFY PASSWORD** request is successful, do not infer that the user ID could also be used to sign on in the CICS region with the **EXEC CICS SIGNON** command. For example, a password verification request cannot identify the following problems:

- The user ID's connections to groups have been revoked.
- The user ID is not authorized to access the CICS address space (identified by the APPLID).
- The user ID is not authorized to use the terminal at which the user is signing on (identified by the TERMINAL class).

If you supply an incorrect password on the **VERIFY PASSWORD** request, the invalid attempt count is increased for the user ID. If you supply multiple incorrect passwords on successive **VERIFY PASSWORD** requests, the user ID may be revoked by the external security manager. CICS issues message DFHXS1201 when you supply an incorrect password on a **VERIFY PASSWORD** request. When you supply a correct password following one or more invalid attempts, CICS issues message DFHXS1206, which includes a count of the invalid attempts.

Unlike the **EXEC CICS SIGNON** command, the **VERIFY PASSWORD** command does not depend upon the principal facility, therefore it can be issued in non-terminal environments such as web applications.

CICS enforces a full verification request at the first time each day that a user ID that is used to log on to the CICS region or is verified using a VERIFY PASSWORD command. The full verification request records the date and time of last access for the user ID, and writes user statistics. A full verification is also made if an incorrect password is entered, and in the next successful request. In other cases, the VERIFY PASSWORD uses a fastpath method to verify the password. For details of the SAF interfaces used, see CICS security control points.

Note: In the CHANGETIME, LASTUSETIME, and EXPIRYTIME options, the time value returned is in the same format as the **ASKTIME** command, that is, in ABSTIME units. For more information about the ABSTIME format, see ASKTIME. The data can be reformatted as a date and time, in a format specified by the caller, by using the **FORMATTIME** command.

Options

CHANGETIME(*data-area***)**

Returns the date and time the password was last changed, in ABSTIME units.

When the external security manager is RACF, the time is shown as midnight.

DAYSLEFT(*data-area***)**
Returns the number of days from now until the password expires, in a halfword binary field. If the password is non-expiring, -1 is returned.

ESMREASON(*data-area***)**
Returns the reason code, in a fullword binary field, that CICS receives from the external security manager.

If the external security manager is RACF, this field is the RACF reason code.

The external security manager does not always return response and reason codes to CICS. Make sure that you check the EIBRESP and EIBRESP2 values returned by this command in addition to checking the ESMRESP and ESMREASON values.

ESMRESP(*data-area***)**
Returns the response code, in a fullword binary field, that CICS receives from the external security manager.

If the external security manager is RACF, this field is the RACF return code.

The external security manager does not always return response and reason codes to CICS. Make sure that you check the EIBRESP and EIBRESP2 values returned by this command in addition to checking the ESMRESP and ESMREASON values.

EXPIRYTIME(*data-area***)**
Returns the date and time the password will expire, in ABSTIME units.

When the external security manager is RACF, the time is shown as midnight.

INVALIDCOUNT(*data-area***)**
Returns the number of times, in a halfword binary field, that an invalid password was entered for this user.

LASTUSETIME(*data-area***)**
Returns the data and time this user ID was last accessed, in ABSTIME units.

PASSWORD(*data-value***)**
Specifies the password, 8 characters, that you want the external security manager to check for the specified user ID. The other data is not returned if the password is not valid.

USERID(*data-value***)**
Specifies the user ID, 8 characters, of the user whose password is to be checked.

Conditions

16 INVREQ
RESP2 values:

13 There is an unknown return code in ESMRESP from the external security manager.

18 The CICS external security manager interface is not initialized.

29 The external security manager is not responding.

32 The user ID field contains a blank character in an invalid position.

Default action: terminate the task abnormally.

70 NOTAUTH
RESP2 values:

2 The supplied password is wrong. If the external security manager is RACF, the revoke count maintained by RACF is incremented.

However, if ESM RESP = 24, the revoke count is not incremented.

3 A new password is required.

19 The user ID is revoked.

20 The user's connection to their default group has been revoked.

ESM RESP values:

24 The RACROUTE REQUEST=VERIFY(X) was failed by an ICHRIX01 installation exit routine.

Default action: terminate the task abnormally.

69 USERIDERR

RESP2 values:

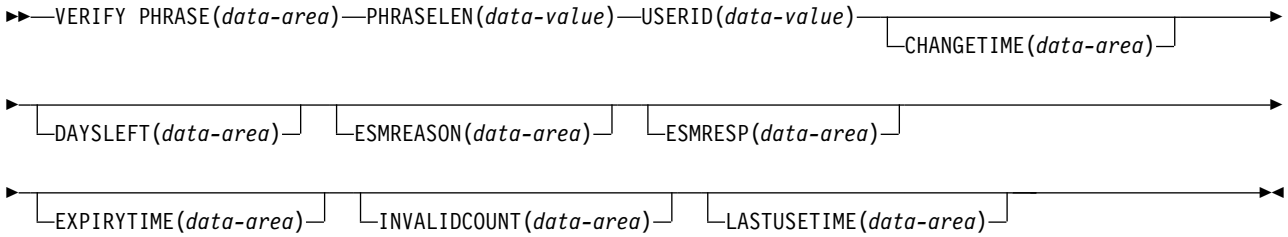
8 The user ID is not known to the external security manager.

Default action: terminate the task abnormally.

VERIFY PHRASE

Verify that a password or password phrase matches the password or password phrase recorded by an external security manager.

VERIFY PHRASE



Conditions: INVREQ, LENGERR, NOTAUTH, USERIDERR

This command is threadsafe.

Description

Use the **VERIFY PHRASE** command to check that a password or password phrase matches the password or password phrase recorded by an external security manager for a user ID. The command returns the values recorded by the external security manager for the password or password phrase. This process is called password verification.

Attention: To ensure that passwords are not revealed in system or transaction dumps, clear the password or password phrase fields on the EXEC CICS commands that have a password or password phrase option as soon as possible after use.

A user ID can have both a standard password and a password phrase. If the length of the phrase as specified by PHRASELEN is between 1 and 8 characters, it is treated as a standard password and the external security manager checks that the PHRASE value matches the password recorded by the external security manager for the user ID. If the length is between 9 and 100 characters, it is treated as a password phrase and the external security manager checks that the PHRASE value matches the password phrase recorded for the user ID.

Although the expiry interval is the same for passwords and password phrases, because they are changed independently, there are separate values for the CHANGETIME and DAYSLEFT options. The values returned for these two parameters depend on whether a valid password or a password phrase is used in the **VERIFY PHRASE** command.

If a **VERIFY PHRASE** request is successful, do not infer that the user ID could also be used to sign on in the CICS region with the **EXEC CICS SIGNON** command. For example, a password verification request cannot identify the following problems:

- The user ID's connections to groups have been revoked.
- The user ID is not authorized to access the CICS address space (identified by the APPLID).
- The user ID is not authorized to use the terminal at which the user is signing on (identified by the TERMINAL class).

Unlike the **EXEC CICS SIGNON** command, the **VERIFY PHRASE** command does not depend upon the principal facility, therefore it can be issued in non-terminal environments such as web applications.

CICS enforces a full verification request at the first time each day that a user ID that is used to log on to the CICS region or is verified using a VERIFY PASSWORD command. The full verification request records the date and time of last access for the user ID, and writes user statistics. A full verification is also made if an incorrect password is entered, and in the next successful request. In other cases, the VERIFY PASSWORD uses a fastpath method to verify the password. For details of the SAF interfaces used, see CICS security control points.

Note: In the CHANGETIME, LASTUSETIME, and EXPIRYTIME options, the time value returned is in the same format as the **ASKTIME** command, that is, in ABSTIME units. For more information about the ABSTIME format, see ASKTIME. The data can be reformatted as a date and time, in a format specified by the caller, by using the **FORMATTIME** command.

Options

CHANGETIME(*data-area***)**
Returns the date and time the password or password phrase was last changed, in ABSTIME units.

When the external security manager is RACF, the time is shown as midnight.

DAYSLEFT(*data-area***)**
Returns the number of days from now until the password or password phrase expires, in a halfword binary field. If the password or password phrase does not expire, a value of -1 is returned. If a user has a password or password phrase that does not expire, DAYSLEFT has no meaning and is shown as -1.

ESMREASON(*data-area***)**
Returns the reason code, in a fullword binary field, that CICS receives from the external security manager.

If the external security manager is RACF, this field is the RACF reason code.

The external security manager does not always return response and reason codes to CICS. Make sure that you check the EIBRESP and EIBRESP2 values returned by this command in addition to checking the ESMRESP and ESMREASON values.

ESMRESP(*data-area***)**
Returns the response code, in a fullword binary field, that CICS receives from the external security manager.

If the external security manager is RACF, this field is the RACF return code.

The external security manager does not always return response and reason codes to CICS. Make sure that you check the EIBRESP and EIBRESP2 values returned by this command in addition to checking the ESMRESP and ESMREASON values.

EXPIRYTIME(*data-area***)**
Returns the date and time the password will expire, in ABSTIME units.

When the external security manager is RACF, the time is shown as midnight. If a user has a password or password phrase that does not expire, EXPIRYTIME has no meaning and is shown as -1.

INVALIDCOUNT(*data-area***)**
Returns the number of times, in a halfword binary field, that an invalid password or password phrase was entered for this user.

LASTUSETIME(*data-area***)**
Returns the data and time this user ID was last accessed, in ABSTIME units.

PHRASE(*data-area***)**
Specifies a 1- to 8-character password or a 9- to 100-byte password phrase required by the ESM. The other data is not returned if the phrase is not valid.

PHRASELEN(*data-value***)**
Specifies the length of the password or password phrase as a fullword binary value.

USERID(*data-value***)**
Specifies the user ID associated with the password or password phrase that to be checked.

Conditions

16 INVREQ
RESP2 values:

13 The external security manager has issued an unknown return code in ESMRESP.

18 The CICS external security manager interface is not initialized.

29 The external security manager is not responding.

32 The USERID field contains a blank character in an invalid position.

Default action: terminate the task abnormally.

22 LENGERR
RESP2 values:

1 PHRASELEN was out-of-range.

70 NOTAUTH
RESP2 values:

2 The supplied password or password phrase is wrong. If the external security manager is RACF, the revoke count maintained by RACF is incremented.

However, if ESM RESP = 24, the revoke count is not incremented.

3 A new password or password phrase is required.

19 The user ID is revoked.

20 The user's connection to their default group has been revoked.

ESM RESP values:

24 The RACROUTE REQUEST=VERIFY(X) was failed by an ICHRIX01 installation exit routine.

Default action: terminate the task abnormally.

69 USERIDERR
RESP2 values:

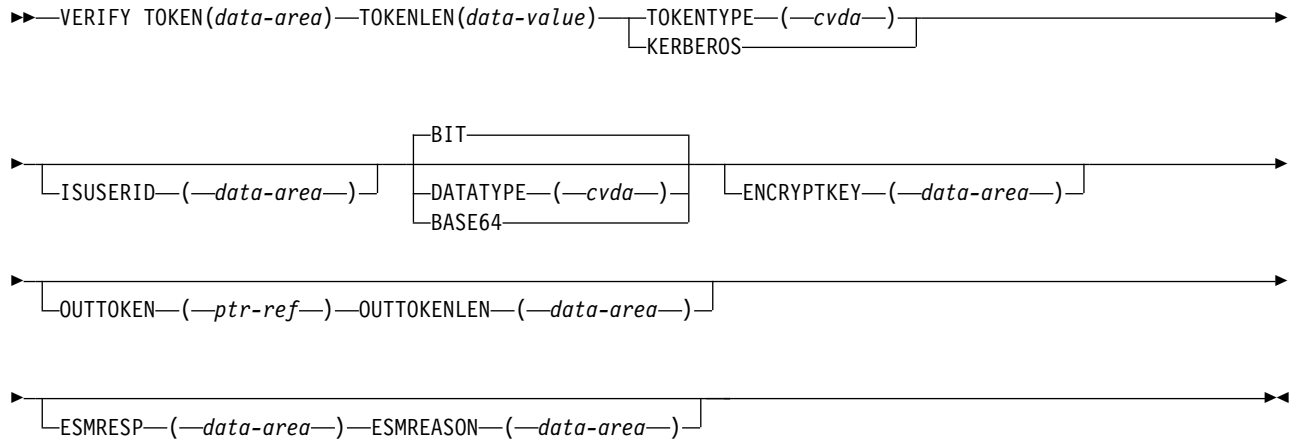
8 The user ID is not known to the external security manager.

Default action: terminate the task abnormally.

VERIFY TOKEN

Verify that a Kerberos token is valid, as determined by an external security manager, and optionally allow the caller to extract the RACF user ID that is associated with the principal in the Kerberos token.

VERIFY TOKEN



Conditions: INVREQ, LENGERR, NOTAUTH

This command is threadsafe.

Description

Use the **VERIFY TOKEN** command to verify that a Kerberos token is valid, as determined by an external security manager. The command optionally returns the user ID of a Kerberos principal that is associated with the token. If the Kerberos token indicates mutual authentication is in use, the command returns a Kerberos output token.

The **VERIFY TOKEN** command uses the z/OS Security Server to verify that the token is a valid Kerberos token and that it can be used by the CICS region. If **ISUSERID** is specified, the user ID of the Kerberos principal for the token is obtained.

The **VERIFY TOKEN** command does not depend upon the principal facility. Therefore, it can be issued in non-terminal environments, for example to provide authentication for web services.

If the external security manager is RACF, the CICS region in which the command is run must be authorized by RACF so that the Kerberos principal for the token can be obtained. For more information, see Configuring RACF for Kerberos. This RACF authorization is required whether **ISUSERID** is specified.

If **ENCRYPTKEY** is specified, **TOKEN** must be a Kerberos token that is obtained from a Security Token Service that supports message confidentiality.

Note: When **ENCRYPTKEY** is specified, an external security manager such as RACF is called by using the **z/OS Security Server RACF Callable Services** > **R_GenSec function code 2 GSS-API** > **Accept a GSS-API security context**. This **R_Gensec** subfunction can be called once for a particular Kerberos token. As a result the **VERIFY TOKEN** command with the **ENCRYPTKEY** option can be called once for a particular Kerberos token.

If **TOKEN** is a Kerberos token that indicates mutual authentication is in use, **OUTTOKEN** and **OUTTOKENLEN** options must be specified. In this case, **OUTTOKEN** returns an

output token that is obtained from an external security manager such as z/OS Security Server, and **OUTTOKENLEN** returns the length of the output token. The calling program should return the output token to the system that supplied the Kerberos token, to allow authentication of the CICS region.

If **TOKEN** is a Kerberos token that does not indicate mutual authentication is in use, and if **OUTTOKEN** and **OUTTOKENLEN** are specified, the values returned in these options are zero.

Note: When **OUTTOKEN** is specified, the **VERIFY TOKEN** command cannot be used by the same long running task repeatedly. Storage that is used for the output token is freed when the task terminates.

For more information about a security failure of this command, see the error messages that are written to destination CSCS.

Options

DATATYPE

Specifies the type of data in the TOKEN and OUTTOKEN options. CVDA values are as follows:

BIT Bit data. This is the default value.

BASE64

Base64 encoded character data. The acceptable characters are A-Z a-z 0-9 + / =.

If your character data is not in a US EBCDIC compatible character CCSID, you must convert it. You can use the CONTAINER API to do the conversion.

ENCRYPTKEY(*data-area***)**

This is the 4-byte token to be used to encrypt a returned PassTicket.

ESMREASON(*data-area***)**

Returns the reason code, in a fullword binary field, that CICS receives from the external security manager.

If the external security manager is RACF, this field is the RACF reason code.

If the external security manager is RACF, and **ENCRYPTKEY** is used, the possible values are the RACF reason codes that are supplied by the **z/OS Security Server RACF Callable Services** > **R_GenSec function code 2 GSS-API** > **Accept a GSS-API security context**. See Subfunction codes in z/OS Security Server RACF Callable Services. For an explanation of a reason code, see R_GenSec Return and reason codes z/OS Security Server RACF Callable Services and Status codes in z/OS Integrated Security Services Network Authentication Service Administration.

The external security manager does not always return response and reason codes to CICS. Make sure that you check the EIBRESP and EIBRESP2 values that are returned by this command in addition to checking the ESMRESP and ESMREASON values.

ESMRESP(*data-area***)**

Returns the response code, in a fullword binary field, that CICS receives from the external security manager.

If the external security manager is RACF, this field is the RACF return code.

The external security manager does not always return response and reason codes to CICS. Make sure that you check the EIBRESP and EIBRESP2 values that are returned by this command in addition to checking the ESMRESP and ESMREASON values.

ISUSERID(*data-area***)**
Returns an 8-byte user ID of a Kerberos principal that is associated with the token.

OUTTOKEN(*ptr-ref***)**
Returns the address in 31-bit user-key storage of the output token that is obtained from an external security manager.

OUTTOKENLEN(*data-area***)**
Returns the length of an output token, in a fullword binary field with a maximum value of 1024.

TOKEN(*data-area***)**
A token that is obtained from a Security Token Service (STS).

TOKENLEN(*data-value***)**
The length of the token as a fullword binary value.

TOKENTYPE(*cvda***)**
Indicates the type of token.

KERBEROS
The token is a Kerberos token.

Conditions

16 INVREQ
RESP2 values are as follows:

13 The external security manager issues an unknown return code in **ESMRESP**.

18 The CICS external security manager interface is not initialized.

29 The external security manager is not responding.

31 A CVDA value other than KERBEROS was specified for **TOKENTYPE**.

32 A CVDA value other than BASE64 or BIT was specified for **DATATYPE**.

36 A data-type of BASE64 was specified, but **TOKEN** does not contain BASE64 data.

40 The key distribution center is not started or is terminating.

41 The key distribution center is not responding.

47 The external security manager does not have a user ID defined for the Kerberos principal that is associated with the token.

50 The data that is specified in **TOKEN** is not a Kerberos token.

51 The Kerberos token does not support message confidentiality.

52 The Kerberos token indicates mutual authentication is in use, but **OUTTOKEN** and **OUTTOKENLEN** have not been specified on the command.

22 LENGERR
RESP2 values are as follows:

45 The length of the Kerberos token exceeds the maximum value of 65535.

Default action: terminate the task abnormally.

70 NOTAUTH
RESP2 values are as follows:

20
The external security manager does not authorize the request to verify the token. See the error messages that are written to destination CSCS.

42 A Kerberos request cannot be completed because the associated ticket expired.

43 The authenticator expired.

Default action: terminate the task abnormally.

WAIT CONVID (APPC)

Ensure that accumulated data is transmitted on an APPC mapped conversation.

WAIT CONVID (APPC)



Conditions: INVREQ, NOTALLOC

Description

WAIT CONVID allows an application program to ensure that any accumulated application data and control indicators from a SEND command, or the results of a CONNECT PROCESS command, are transmitted to the partner transaction.

Options

CONVID(*name*)
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

STATE(*cvda*)
gets the state of the current conversation. The cvda values returned by CICS are:

- ALLOCATED
- CONFFREE
- CONFRECEIVE
- CONFSEND
- FREE
- PENDFREE
- PENDRECEIVE
- RECEIVE
- ROLLBACK
- SEND

- SYNCFREE
- SYNCRECEIVE
- SYNCSEND

Conditions

16 INVREQ
RESP2 values:

200 Command not supported for distributed program link when it refers to the principal facility.

also occurs (RESP2 not set) if he command is used on a conversation that is not using the EXEC CICS interface or that is not a mapped conversation.

Default action: terminate the task abnormally.

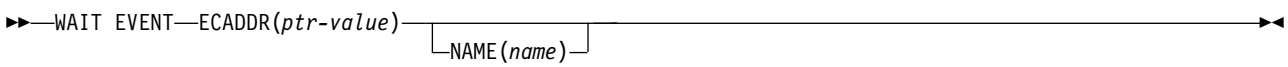
61 NOTALLOC
occurs if the CONVID value does not relate to a conversation that is owned by the application.

Default action: terminate the task abnormally.

WAIT EVENT

Wait for an event to occur.

WAIT EVENT



Condition: INVREQ

Note for dynamic transaction routing: Using this command could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. See the Affinity for more information about transaction affinities.

Description

WAIT EVENT synchronizes a task with the completion of an event initiated by the same task or by another task. The event would normally be the posting, at the expiration time, of a timer-event control area provided in response to a POST command, as described in “POST” on page 349. The WAIT EVENT command provides a method of directly relinquishing control to some other task until the event being waited on is completed.

CICS includes the addresses of all ECBs passed by WAIT EVENT commands of current tasks in the ECBLIST passed by CICS to the WAIT facility when it runs out of work.

A given ECB may not be waited on by more than one task at the same time. If this rule is not followed and the ECBLIST passed by CICS on the MVS WAIT contains duplicate ECB addresses, MVS abends CICS.

Make sure that asynchronous cross memory post (posting completion of an event in an address space other than the user's own) is not used more frequently than

necessary. Large numbers of cross memory posts can consume excessive amounts of system resources.

Options

ECADDR(*ptr-value***)**
specifies the address of the timer-event control area that must be posted before task activity can be resumed.

NAME(*name***)**
specifies the symbolic name, 1–8 alphanumeric characters, that is returned in SUSPENDVALUE or HVALUE, when a task issues WAIT EVENT and is the subject of an INQUIRE TASK command or a CEMT INQ TASK.

Conditions

16 INVREQ
RESP2 values:

2 The ECB address is a null pointer, (X'00000000') or (X'FF000000').

3 The specified event control area address is above the 16MB line for programs executing in 24-bit mode.

4 The event control area address is not aligned on a fullword boundary.

6 The timer-event control area specified on a WAIT EVENT is in user-key task-lifetime storage, and is inaccessible to another transaction. This condition can only occur if the storage for the timer-event control area is obtained other than by a POST command, and is for posting as an ECB by some other task on completion of an event.

Note: CICS obtains storage for a timer-event control area in response to a POST command (and which can be used in conjunction with the WAIT EVENT command) from a shared subpool in user-key storage. This ensures that timer-event control areas are in shared storage and, when referenced by a subsequent WAIT EVENT command, do not fail with an INVREQ.

Default action: terminate the task abnormally.

Examples

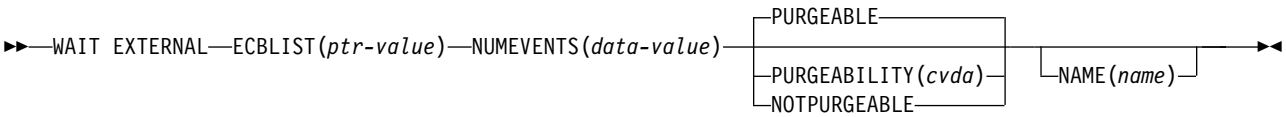
The following example shows you how to suspend the processing of a task until the specified event control area is posted:

```
EXEC CICS WAIT EVENT ECADDR(PVALUE)
```

WAIT EXTERNAL

Synchronize events.

WAIT EXTERNAL



Condition: INVREQ

This command is threadsafe.

Note for dynamic transaction routing: Using this command could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. See Affinity for more information about transaction affinities.

Description

WAIT EXTERNAL waits for events that post MVS-format ECBs. The command causes the issuing task to be suspended until one of the ECBs has been posted, that is until one of the events has occurred. The task can wait on one or more ECBs. If it waits on more than one, it is dispatchable as soon as one of the ECBs is posted. You must ensure that each ECB is cleared (set to binary zeros) no later than the earliest time it could be posted. CICS cannot do this for you. If you wait on an ECB that has been previously posted and not subsequently cleared, your task is not suspended and continues to run as though the WAIT EXTERNAL had not been issued.

CICS uses extended ECBs and the MVS POST exit mechanism for ECBs passed by WAIT EXTERNAL; therefore do not use WAIT EXTERNAL unless you are sure that the ECBs are not posted by any method other than the MVS POST service or the standard 'optimized post' logic using a compare-and-swap (CS) instruction. Note that the standard 'optimized post' logic is only applicable when the ECB is not waiting, that is when the wait bit X'80' is not on.

If a WAIT EXTERNAL ECB is hand posted, for example by another task moving a value into the ECB, unpredictable errors occur. If there is any possibility of hand posting, use the WAITCICS command. Use WAIT EXTERNAL whenever possible, because it usually has less overhead.

A given ECB must not be waited on by more than one task at the same time. If this rule is not followed, the second task to wait on the ECB gets an INVREQ condition.

Make sure that asynchronous cross memory post (posting completion of an event in an address space other than the user's own) is not used more frequently than necessary. Large numbers of cross memory posts can consume excessive amounts of system resources.

Options

ECBLIST(*ptr-value***)**
is a pointer to a list of addresses of MVS-format ECBs representing events. Both the ECBLIST and the ECBs can be above the 16MB line, that is they can be 31-bit addresses. Each ECB must be fullword aligned. Null (X'00000000' and X'FF000000') ECB addresses are ignored.

NAME(*name***)**
specifies the symbolic name, 1–8 alphanumeric characters, that is returned in SUSPENDVALUE or HVALUE, when a task issues WAIT EXTERNAL and is the subject of an INQ TASK command or a CEMT INQ TASK.

NUMEVENTS(*data-value***)**
is the number of such events, corresponding to the number of addresses in the ECBLIST. The field is fullword binary. When NUMEVENTS is specified as 1, ECBLIST must still be an address that points to a list containing just one ECB address.

PURGEABILITY(*cvda***)**
determines the outcome of:

- An attempt to perform a deadlock timeout
- A SET TASK PURGE | FORCEPURGE command
- A CEMT SET TASK PURGE | FORCEPURGE

on the issuing task while it is waiting. The values passed to CICS are PURGEABLE (the default value), or NOTPURGEABLE. The outcome is:

Function	**PURGEABLE**	**NOTPURGEABLE**
`DTIMOUT` expired	Abend AEXY	No effect

`CEMT SET TASK PURGE`
`EXEC CICS SET TASK`
`PURGE`
	Abend AEXY	No effect

`CEMT SET TASK`
`FORCEPURGE`
`EXEC CICS SET TASK`
`FORCEPURGE`
	Abend AEXY	Abend AEXY

See Recovery for transactions for information about DTIMOUT and SET TASK PURGE | FORCEPURGE.

Conditions

16 INVREQ
RESP2 values: CVDA values are:

1 An ECB is not valid, for example the ECB is not fullword aligned.

2 The ECB address is a null pointer, (X'00000000') or (X'FF000000').

3 NUMEVENTS is not a positive number.

4 PURGEABILITY is specified with an incorrect CVDA.

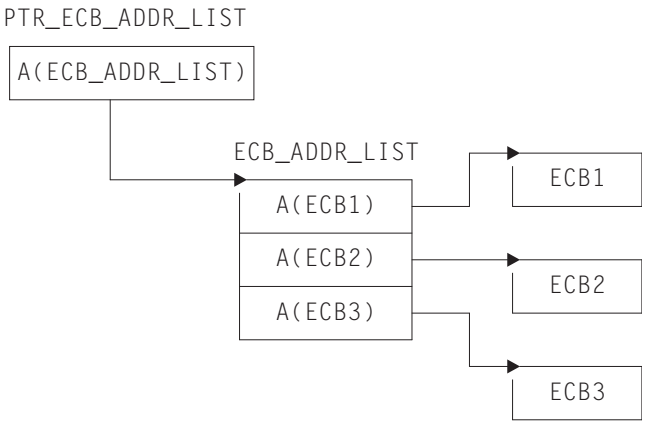
5 No valid ECBs have been found in the list, because either the ECBLIST address is not valid or all the ECB addresses are not valid.

The ECBs specified are in read-only storage.

Default action: terminate the task abnormally.

Examples

The following figure shows how to use the ECBLIST parameter to point to a list of ECB addresses that in turn point to individual ECBs. Note that the ECBLIST variable is a pointer pointing to the first address of the list.



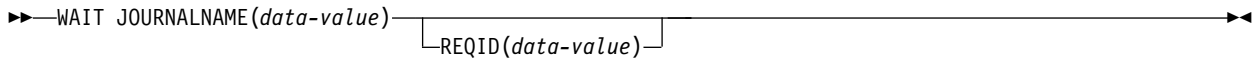

```
DCL
   ECB1        FIXED BIN(31),      /* actual ecb */
   ECB2        FIXED BIN(31),      /* actual ecb */
   ECB3        FIXED BIN(31);      /* actual ecb */
DCL                /* list of ecb addresses */
   1  ECB_ADDR_LIST,
      2  ECB_ADDR(3)  PTR;
DCL                /* ptr to each addr list */
   PTR_ECB_ADDR_LIST  PTR;
ECB_ADDR(1) = ADDR(ECB1);
ECB_ADDR(2) = ADDR(ECB2);
ECB_ADDR(3) = ADDR(ECB3);
                           /* set up pointer */
PTR_ECB_ADDR_LIST = ADDR(ECB_ADDR_LIST);
/* PTR_ECB_ADDR_LIST = ADDR(ECB_ADDR(1));
                                (alternative) */
 EXEC CICS WAIT EXTERNAL
                ECBLIST(PTR_ECB_ADDR_LIST)
                NUMEVENTS(3)
                PURGEABLE
```

Figure 3. ECBLIST option, EXEC CICS WAIT EXTERNAL

WAIT JOURNALNAME

Synchronize the task with journal output being written to a system logger log stream.

WAIT JOURNALNAME



Conditions: IOERR, JIDERR, NOTOPEN

This command is threadsafe.

Description

WAIT JOURNALNAME synchronizes the task with the output of one or more journal records that have been created but whose output has been deferred; that is, with asynchronous journal output requests.

The journal records may already be written out from the journal buffer area to the corresponding system logger log stream, or the system logger output operation may be in progress. If the log stream output operation has already been completed, control returns immediately to the requesting task; if not, the requesting task waits until the operation has been completed.

If the requesting program has made a succession of successful asynchronous output requests to the same journal, it is necessary to synchronize on only the last of these requests to ensure that all of the journal records have been output to the system logger log stream. This may be done either by issuing a stand-alone WAIT JOURNALNAME command, or by making the last output command itself synchronous (by specifying the WAIT option in the WRITE JOURNALNAME command).

Options

JOURNALNAME(data-value)
specifies a 1– to 8– character journal name identifying the journal on which the task is to wait for synchronization. The name must be a journal name known to CICS.

To issue the wait against the CICS system log, specify DFHLOG as the journal name.

To issue the wait against journals defined using the journal numbering convention, as in file resource definitions, specify the name as DFHJ*nn*, where nn is the journal number in the range 1 to 99.

Note: Specifying DFHJ01 on this command refers to a user journal, *not* the system log.

REQID(data-value)
specifies, in a fullword binary variable, the token that refers to a journal record that has been created but possibly not yet written out. The token is returned by CICS from a previous WRITE JOURNALNAME command issued by this task.

If you do not specify REQID, the task is synchronized with the output of the current buffer for the journal specified by JOURNALNAME.

Conditions

17 IOERR
a journal record was not output because of an irrecoverable error condition returned by the system logger or by SMF.

Default action: If the log is the system log, CICS either quiesces or abends. If it is a general log, the task is terminated abnormally.

43 JIDERR
occurs in either of the following situations:

- The specified journal name is not known in the CICS region.
- The specified journal name refers to a DASD-only log stream to which a CICS region in another MVS image is currently connected.

Default action: terminate the task abnormally.

19 NOTOPEN
occurs in any of the following situations:

- The command cannot be satisfied because the user explicitly disabled the specified journal.
- A wait request is issued against a journal that has not previously been written to.
- This journal was defined using a model that maps it into the logstream used by the system log for this system. This error is not detected when trying to connect to the logstream. A definition for the journal will be installed and set to failed.

Default action: terminate the task abnormally.

70 NOTAUTH
A resource security check has failed on the JOURNALNAME (data-value).

Default action: terminate the task abnormally.

Examples

The following example shows how to request synchronization with the output of journal records written to a user journal named 'ACCOUNTS'

```
EXEC CICS WAIT JOURNALNAME('ACCOUNTS')
     REQID(RECTOKEN)
```

WAIT JOURNALNUM

Synchronize with journal output.

This command is supported for compatibility with earlier releases of CICS. It is superseded by the WAIT JOURNALNAME command, which you are recommended to use instead.

The syntax is the same as for WAIT JOURNALNAME except that JOURNALNUM specifies a numeric instead of a character value. The numeric value, nn, is in the range 01-99 and corresponds to journal name DFHJnn.

This command is threadsafe.

WAIT SIGNAL

Suspend task on a logical unit.

WAIT SIGNAL



Conditions: NOTALLOC, SIGNAL, TERMERR

Description

WAIT SIGNAL, for a principal facility only, suspends a task until a SIGNAL condition occurs. Some logical units can interrupt the normal flow of data to the application program by a SIGNAL data-flow control command to CICS, signaling an attention, that in turn causes the SIGNAL condition to occur.

The HANDLE CONDITION SIGNAL command causes a branch to a user routine when an attention is received.

The logical units for which you can code a WAIT SIGNAL command are:

- LUTYPE4
- LUTYPE6.1
- 3600 (3601)
- 3767 interactive
- 3770 batch
- 3790 full-function

Conditions

61 NOTALLOC
occurs if the principal facility of the task is not a terminal.

Default action: terminate the task abnormally.

24 SIGNAL
occurs when the data-flow control command has been received from the principal facility.

EIBSIG is always set when an inbound signal is received.

Default action: ignore the condition.

81 TERMERR
occurs for a terminal-related error.

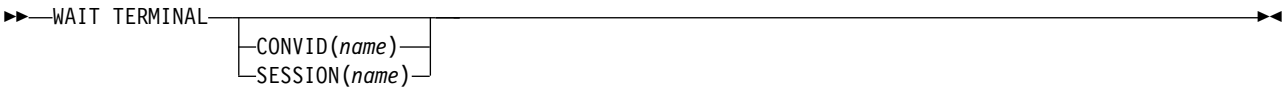
A CANCEL TASK request by a user node error program (NEP) may cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program (CSNE) handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

WAIT TERMINAL

Ensure terminal operation has completed.

WAIT TERMINAL



Conditions: EOC, INVREQ, NOTALLOC, SIGNAL, TERMERR

Description

WAIT TERMINAL ensures that terminal operation has completed.

Options

CONVID(*name***)**
identifies the conversation to which the command relates. The 4-character name identifies either the token returned by a previously executed ALLOCATE command in EIBRSRCE in the EIB, or the token representing the principal session (returned by a previously executed ASSIGN command).

SESSION(*name***)**
specifies the symbolic identifier (1–4 characters) of a session TCTTE. This option specifies the alternate facility to be used. If both this option and CONVID are omitted, the principal facility for the task is used.

Conditions

06 EOC
occurs when a request/response unit (RU) is received with end-of-chain-indicator set. Field EIBEOC also indicates this condition.

Default action: ignore the condition.

16 INVREQ
RESP2 values:

200 A distributed program link server application specified the function-shipping session (its principal facility) on the CONVID option.

Default action: terminate the task abnormally.

61 NOTALLOC
occurs if the facility specified in the command is not owned by the application.

Default action: terminate the task abnormally.

24 SIGNAL
occurs when an inbound SIGNAL data-flow control command is received from a logical unit or session. EIBSIG is always set when an inbound signal is received.

Default action: ignore the condition.

81 TERMERR
occurs for a terminal-related error.

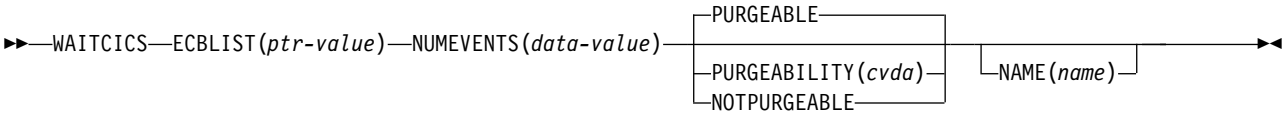
A CANCEL TASK request by a user node error program (NEP) can cause this condition if the task has an outstanding terminal control request active when the node abnormal condition program handles the session error.

Default action: terminate the task abnormally with abend code ATNI.

WAITCICS

Synchronize events.

WAITCICS



Condition: INVREQ

Note for dynamic transaction routing: Using this command could create inter-transaction affinities that adversely affect the use of dynamic transaction routing. See Affinity for more information about transaction affinities.

Description

WAITCICS waits for events that post MVS-format ECBs. The command causes the issuing task to be suspended until one of the ECBs has been posted, that is until one of the events has occurred. The task can wait on one or more ECBs. If it waits on more than one, it is dispatchable as soon as one of the ECBs is posted. You must ensure that each ECB is cleared, set to binary zeros, no later than the earliest time it could be posted. CICS cannot do this for you. If you wait on an ECB that has been previously posted and not subsequently cleared, your task is not suspended and continues to run as though the WAITCICS had not been issued.

CICS includes the addresses of all ECBs passed by WAITCICS commands of current tasks in the ECBLIST passed by CICS to the MVS WAIT facility when it runs out of work. Such ECBs can be posted using the MVS POST facility or by hand posting. Hand posting could, for example, be done by moving an appropriate value into the ECB. If hand posting is definitely not going to be used, it is preferable to use WAIT EXTERNAL.

A given ECB may not be waited on by more than one task at the same time. If this rule is not followed and the ECBLIST passed by CICS on the MVS WAIT contains duplicate ECB addresses, MVS abends CICS.

Options

ECBLIST(*ptr-value***)**
is a pointer to a list of addresses of MVS-format ECBs representing events. Both the ECBLIST and the ECBs can be above the 16MB line, that is they can be 31-bit addresses. Each ECB must be fullword aligned. Null (X'00000000' and X'FF000000') ECB addresses are ignored.

NAME(*name*)
specifies the symbolic name, 1–8 alphanumeric characters, as the reason for the wait. The value you specify is returned in the SUSPENDVALUE or HVALUE option respectively of the EXEC CICS INQ TASK or CEMT INQ TASK commands.

NUMEVENTS(*data-value*)
is the number of such events, corresponding to the number of addresses in the ECBLIST. The field is fullword binary. When NUMEVENTS is specified as one, ECBLIST must still be an address that points to a list containing just one ECB address.

PURGEABILITY(*cvda*)
causes the issuing task to be suspended until one of the ECBs has been posted; that is, until one of the events has occurred. The values passed to CICS are PURGEABLE (the default value), or NOTPURGEABLE. The field is fullword binary. If while this task is waiting another function attempts to purge it, the result is as follows:

Function	PURGEABLE	NOTPURGEABLE
DTIMOUT expired	Abend AEXY	No effect
CEMT SET TASK PURGE EXEC CICS SET TASK PURGE	Abend AEXY	No effect
CEMT SET TASK FORCEPURGE EXEC CICS SET TASK FORCEPURGE	Abend AEXY	Abend AEXY

See Recovery for transactions for information about DTIMOUT and SET TASK for information about SET TASK PURGE|FORCEPURGE.

Conditions

16 INVREQ
RESP2 values:

1 An ECB is not valid, for example the ECB is not fullword aligned.

3 NUMEVENTS is not a positive number.

4 PURGEABILITY is specified with an incorrect CVDA.

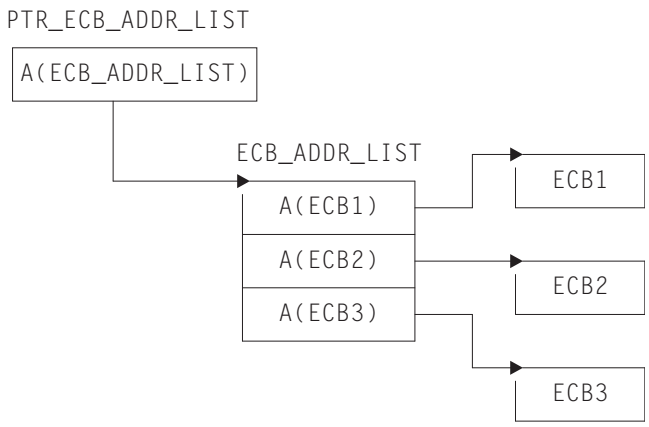
5 No valid ECBs have been found in the list, because either the ECBLIST address is not valid, or all the ECB addresses are not valid.

The ECBs specified are in read-only storage.

Default action: terminate the task abnormally.

Examples

The following figure shows how to use the ECBLIST parameter to point to a list of ECB addresses that in turn point to individual ECBs. Note that the ECBLIST variable is a pointer pointing to the first address of the list.




```
DCL
   ECB1        FIXED BIN(31),      /* actual ecb */
   ECB2        FIXED BIN(31),      /* actual ecb */
   ECB3        FIXED BIN(31);      /* actual ecb */
DCL               /* list of ecb addresses */
   1  ECB_ADDR_LIST,
      2  ECB_ADDR(3)  PTR;
DCL                /* ptr to each addr list */
   PTR_ECB_ADDR_LIST  PTR;
ECB_ADDR(1) = ADDR(ECB1);
ECB_ADDR(2) = ADDR(ECB2);
ECB_ADDR(3) = ADDR(ECB3);
                          /* set up pointer */
PTR_ECB_ADDR_LIST = ADDR(ECB_ADDR_LIST);
/* PTR_ECB_ADDR_LIST = ADDR(ECB_ADDR(1));
                              (alternative) */
 EXEC CICS WAITCICS
           ECBLIST(PTR_ECB_ADDR_LIST)
           NUMEVENTS(3)
           PURGEABLE
```

Figure 4. ECBLIST option, EXEC CICS WAITCICS

WEB CLOSE

Completes the application's use of a connection to a server for CICS as an HTTP client.

WEB CLOSE



Conditions: NOTOPEN

This command is threadsafe.

Description

WEB CLOSE notifies CICS that the application program has finished using a client HTTP connection to a server. The session token represents the application's use of the connection. After the application issues the WEB CLOSE command, the specified session token is no longer valid for use. The session token is required to

receive a response from the server and to read the HTTP headers for the response, so the WEB CLOSE command should not be issued until all interaction with the server and with the response that it sends is complete.

When the application program finishes using a client HTTP connection, CICS might or might not close the connection:

- If the connection is still open when you issue the WEB CLOSE command, and it was opened using a URIMAP resource that specified connection pooling, CICS does not close the connection. CICS checks the state of the connection and places it in a pool for reuse.
- If the connection is not suitable for connection pooling when you issue the WEB CLOSE command, because the server or your application program has previously made a request to close the connection, or it was not opened using a suitable URIMAP resource, or it is not in a good state, CICS closes the connection and does not place it in a pool.
- If you do not issue the WEB CLOSE command, CICS closes the connection at end of task. The closed connection cannot be placed in a pool. To enable connection pooling, your application must issue the WEB CLOSE command.

When you issue the WEB CLOSE command, CICS does not request the server to close the connection. If you are not using connection pooling, and CICS will close the connection after the application program has finished, it is good practice to request the server to close the connection by specifying the CLOSESTATUS(CLOSE) option on the final WEB SEND or WEB CONVERSE command. When this option is specified, CICS writes a Connection: close header on the request, or, for a server at HTTP/1.0 level, omits the Connection: Keep-Alive header. The information in the headers means that the server can close its connection immediately after sending the final response, rather than waiting for any further requests before timing out. If you are using connection pooling, do not specify the CLOSESTATUS(CLOSE) option, because closed connections cannot be pooled.

The connection might be closed at the request of the server before the WEB CLOSE command is issued. If you need to test whether the server has requested closure of the connection, use the WEB READ HTTPHEADER command to look for the Connection: close header in the last message from the server.

If the server does request closure of the connection, the data relating to that connection is still kept available within CICS until the WEB CLOSE command is issued. The available data includes the most recent message received from the server, and the parameters used to open the connection (such as the scheme and the host name of the server). When a server has closed the connection, the application program cannot perform the following tasks:

- Send further requests on that connection, using the WEB SEND or WEB CONVERSE commands.
- Write HTTP headers, using the WEB WRITE HTTPHEADER command.

However, the application program can still perform the following tasks:

- Receive a response, using the WEB RECEIVE command.
- Examine HTTP headers, using the WEB READ HTTPHEADER and HTTP header browsing commands.
- Extract connection information, using the WEB EXTRACT command.

When the WEB CLOSE command is issued, the data associated with the application's use of the connection is cleared.

Options

SESSTOKEN*(data-value)*

Specifies the session token, an 8-byte binary value that uniquely identifies an application's use of a client HTTP connection between CICS and a server. This value is returned by a WEB OPEN command for CICS as an HTTP client. When you issue the WEB CLOSE command for the connection identified by the session token, CICS clears the data associated with the application's use of the connection, and makes the session token invalid for further use by the application program. Session tokens explains the use of the session token.

Conditions

19 NOTOPEN

RESP2 values are:

27 Invalid session token.

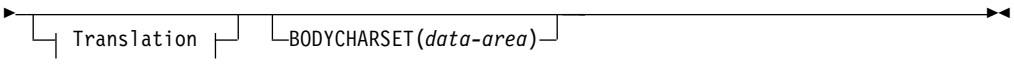
144 One or more of the Web command parameters is invalid.

WEB CONVERSE

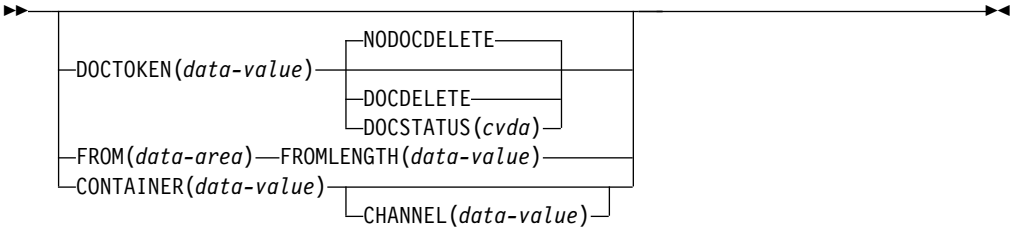
Send an HTTP request by CICS as an HTTP client, and receive a response from the server, using a single command. WEB CONVERSE is an alternative to the WEB SEND and WEB RECEIVE commands for CICS as an HTTP client.

WEB CONVERSE

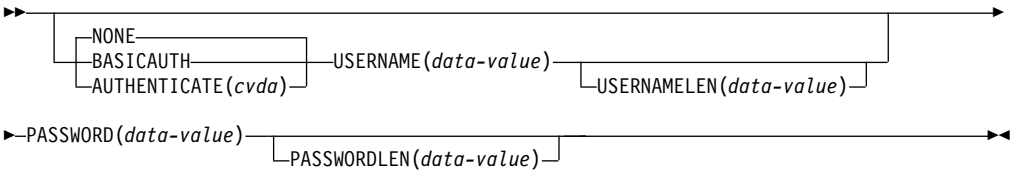




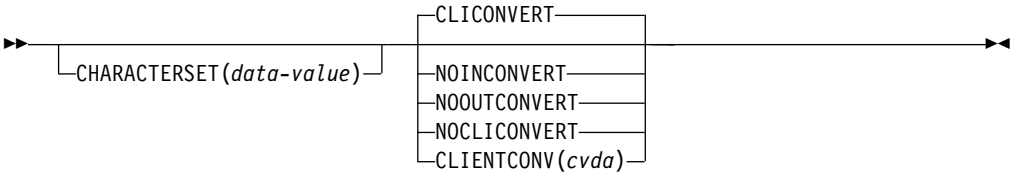
Body



Credentials



Translation



Conditions: CHANNELERR, CONTAINERERR, IOERR, INVREQ, LENGERR, NOTAUTH, NOTFND, NOTOPEN, TIMEDOUT, TOKENERR

This command is threadsafe.

Description

WEB CONVERSE enables an application program to compose and send an HTTP client request, and receive a response from the server. A session token must be included on this command. For guidance on the correct use of the WEB CONVERSE command, see .

- **The HTTP client request** is made using a connection that has been opened using the WEB OPEN command. The WEB CONVERSE command can be used in place of the WEB SEND command to compose and send the request.
- **The response from the server** is received by CICS Web support and passed to the application. The WEB CONVERSE command can be used in place of the WEB RECEIVE command to make the application program wait for and receive the HTTP response. The headers for the HTTP response can be examined separately using the WEB READ HTTPHEADER command or the HTTP header browsing commands.

Note: The RTIMOUT value specified for the transaction that starts the user application indicates the time that the application is prepared to wait to receive the incoming message. (RTIMOUT is specified on the transaction profile definition). When the period specified by RTIMOUT expires, CICS returns a TIMEDOUT response to the application. An RTIMOUT value of zero means that the application is prepared to wait indefinitely. The default setting for RTIMOUT on transaction profile definitions is zero, so it is important to check and change that setting for applications that are making HTTP client requests.

The request can also time out when sending a message to the server. In this case, the deadlock timeout interval specified in the DTIMOUT attribute of the TRANSACTION definition applies, and CICS returns a TIMEDOUT response to the application.

The WEB CONVERSE command does not support chunked transfer-coding for the request, because this requires a sequence of send actions, and the WEB CONVERSE command provides a single send action. If you want to send a chunked message, use the WEB SEND command to send it, and the WEB RECEIVE command to receive it. If the server sends a chunked response, this can be received using the WEB CONVERSE command.

The WEB CONVERSE command cannot be used after the connection to the server has been closed. If you need to test whether the server has requested termination of the connection, use the WEB READ HTTPHEADER command to look for the Connection: close header in the last message from the server.

The WEB CONVERSE command performs a single send action and a single receive action, and it is designed to be used in place of a WEB SEND command and a WEB RECEIVE command. You may use WEB SEND and WEB RECEIVE commands, and WEB CONVERSE commands, in relation to the same connection (that is, with the same SESSTOKEN). However, if you are pipelining requests (that is, sending a sequence of requests without waiting for a response), you must not follow a WEB SEND command with a WEB CONVERSE command. CICS checks at program run time that each WEB SEND command has a subsequent WEB RECEIVE command before any WEB CONVERSE command is issued. For example, if you use the WEB SEND command three times to issue a pipelined sequence of requests, you must use the WEB RECEIVE command three times to receive the responses for those requests, before you can use the WEB CONVERSE command.

Options for sending the HTTP client request

ACTION*(cvda)*

This option is used to specify how the message should be sent out. The CVDA value is as follows:

EXPECT

Makes CICS send an Expect header along with the request line and headers for the request, and await a 100-Continue response before sending the message body to the server. If a response other than 100-Continue is received, CICS informs the application program and cancels the send. If no response is received after a period of waiting, CICS sends the message body anyway.

The Expect header is not supported by servers below HTTP/1.1. If CICS does not yet know the HTTP version of the server, CICS makes an additional request before sending your request, in order to

determine the HTTP version of the server. If the Expect header would not be suitable, CICS sends your request without it.

This option must only be used if your request has a message body.

AUTHENTICATE*(cvda)*
This option allows you to specify user authentication details (credentials), to control access to restricted data. CVDA values are as follows:

NONE
Specifies that there are no restrictions on accessing this data, therefore no credentials are required. This is the default value for AUTHENTICATE.

BASICAUTH
Specifies that HTTP Basic Authentication credentials are required for this session. These details can be supplied within the command or by using the XWBAUTH global user exit.

CHANNEL*(data-value)*
Specifies the name of the channel to which the container belongs. The name of the channel can consist of up to 16 alphanumeric characters, including appropriate punctuation. Leading and embedded blanks are not permitted. If the name is less than 16 characters, it is padded with trailing blanks. You can specify the channel name `DFHTRANSACTION` to use the transaction channel.

If the CONTAINER option is specified, CHANNEL is optional.

If the CHANNEL option is not specified, CICS assumes the current channel.

CHARACTERSET*(data-value)*
Specifies a character set into which CICS translates the entity body of the item sent by the command before sending. The name of the character set can consist of up to 40 alphanumeric characters, including appropriate punctuation. CICS does not support all the character sets named by IANA. HTML coded character sets lists the IANA character sets that are supported by CICS for code page conversion.

To allow code page conversion of the entity body, the CLIENTCONV option must be set (or allowed to default) to CLICONVERT. If the NOCLICONVERT option is specified, code page conversion will not take place. If conversion is requested and CHARACTERSET is not specified, ISO-8859-1 is used as the default character set.

CLOSESTATUS*(cvda)*
Specifies whether or not a Connection header with the "close" connection option (Connection: close) should be included on the request. The default is that the header is not included. CVDA values are as follows:

CLOSE
Makes CICS write a Connection: close header for this request. The header notifies the server that the connection should be closed after the server has sent its response to the request. (For a server at HTTP/1.0 level, CICS achieves the same effect by omitting the Connection: Keep-Alive header.) Do not specify this option if you implemented connection pooling in the URIMAP resource for this connection, because a closed connection cannot be pooled for reuse. Only specify this option if this is your final request to the server and you are not using connection pooling.

NOCLOSE
Means that the Connection: close header is not used for this request. If

the server is identified as HTTP/1.0, CICS sends a Connection header with the "Keep-Alive" connection option (Connection: Keep-Alive), to notify that a persistent connection is intended.

CONTAINER*(data-value)*
Specifies the name of the container where the HTTP body is held, before it is sent to the server. The name of the container can consist of up to 16 alphanumeric characters, including appropriate punctuation. Leading and embedded blanks are not permitted. If the name is shorter than 16 characters, it is padded with trailing blanks.

DOCSTATUS*(cvda)*
Indicates whether the document should be deleted or not deleted during processing of the WEB CONVERSE command. CVDA values are as follows:

DOCDELETE
CICS deletes the document after the document contents are saved for sending. Storage allocated for the document is released immediately. If you make subsequent requests for the document, these generate a TOKENERR response.

NODOCDELETE
CICS does not delete the document during processing of the **WEB CONVERSE** command. This is the default value for DOCSTATUS.

DOCTOKEN*(data-value)*
Specifies the 16-byte binary token of a document to be sent as the message body. The document must be created using the CICS Document interface, using the **EXEC CICS DOCUMENT CREATE**, **INSERT**, and **SET** commands. The FROM option provides an alternative way to create a message body.

FROM*(data-area)*
Specifies a buffer of data which holds the message body. The message body is built by the application program. When you specify the FROM option, use the FROMLENGTH option to specify the length of the buffer of data. The DOCTOKEN option provides an alternative way to create the message body.

There is no set maximum limit for the size of the data area, but its size is limited in practice by storage considerations. Producing an entity body for an HTTP message has more information about storage considerations.

FROMLENGTH*(data-area)*
Specifies the length, as a fullword binary value, of the buffer of data supplied on the FROM option (the message body). It is important to state this value correctly, because an incorrect data length can cause problems for the recipient of the message.

MEDIATYPE*(data-area)*
Specifies the data content of any message body provided, for example `text/xml`. You must specify a 56-byte area for `MEDIATYPE`. The media type is up to 56 alphanumeric characters, including appropriate punctuation, but not spaces. For more information on media types, see IANA media types and character sets. CICS checks that the format of the media type is correct, but does not check the validity of the media type against the data content. CICS uses this information to produce the Content-Type header for the message.

For requests that require a body, you must specify the MEDIATYPE option. There is no default. However, if the required Content-Type header needs to contain spaces or more than 56 characters, the application can provide it using the WEB WRITE HTTPHEADER command. In this case, do not specify the MEDIATYPE option.

The supplied media type is used to determine whether code page conversion is required under the following circumstances:

- If you are sending a message from a buffer, using the FROM option, and the CLIENTCONV and CHARACTERSET options are not specified.
- If you are sending a message from a document, using the DOCTOKEN option, and the CLIENTCONV and CHARACTERSET options are not specified.
- If you are sending a message from a named container, using the CONTAINER option, and either CLICONVERT is specified, or the CLIENTCONV and CHARACTERSET options are not specified.

If the supplied media type is text, the message is converted. If the supplied media type is nontext, the message is not converted.

The MEDIATYPE option is used for both the sending and receiving functions of the WEB CONVERSE command. If it is specified with a value, the value is used to construct the Content-Type header in the request, and the same field is used to receive the media type of the response that is returned by the server. If it is specified without a value, it is used only to receive the media type of the response.

METHOD*(cvda)*

Specifies the HTTP method for the request.

The GET, HEAD, POST, PUT, TRACE, OPTIONS, and DELETE methods are supported by this command. However, some HTTP servers, particularly HTTP/1.0 servers, might not implement all of these methods.

Producing an entity body for an HTTP message has more information about the correct use of methods, including the HTTP versions that apply to each.

CICS bars the sending of a message body for methods where it is inappropriate, and requires it for methods where it is appropriate.CVDA values are as follows:

GET Obtain a resource from the server. A request body is not allowed.

HEAD
Obtain the HTTP headers, but not the response body, for a resource. A request body is not allowed.

POST Send data to a server. A request body is required.

PUT Create or modify a resource on the server. A request body is required.

TRACE
Trace the route of your request to the server. A request body is not allowed.

OPTIONS
Obtain information about the server. A request body is allowed, but there is no defined purpose for the body. If you do use a request body, then you must specify a media type.

DELETE
Delete a resource on the server. A request body is not allowed.

PASSWORD*(data-value)*

Specifies the password associated with the user ID or logon name that is allowed access to this data. The PASSWORD option is required only if the USERNAME option is used. If the specified password is over 8 characters long, it is treated as a password phrase when it is sent to z/OS systems.

PASSWORDLEN*(data-value)*
Specifies the buffer length supplied for the PASSWORD option as a fullword binary variable.

PATH*(data-area)*
Specifies the path information for the specific resource within the server that the application needs to access.

If the URIMAP option was used to specify an existing URIMAP definition on the WEB OPEN command for this connection, the path specified in that URIMAP definition is the default path for the WEB SEND command. In these circumstances, if you do not specify path information on the WEB SEND command, the path from the URIMAP definition is used. If you specify a different path from that given in the URIMAP definition, this overrides the path from the URIMAP definition.

If the URIMAP option was not used on the WEB OPEN command, there is no default path, and you must provide path information. Path information can be extracted from a known URL using the WEB PARSE URL command.

As an alternative to using the PATH option to provide the path information, you can use the URIMAP option on the WEB CONVERSE command to specify a URIMAP definition from which the path information is taken directly.

PATHLENGTH*(data-value)*
Specifies the length of the path, as a fullword binary value. If you are providing path information using the PATH option, you need to specify the PATHLENGTH option. Path length information is returned if you use the WEB PARSE URL command to parse a URL.

QUERYSTRING*(data-area)*
Specifies a query string that is to be supplied to the server as part of the request. You do not need to include a question mark (?) at the beginning of the query string; if you do not include it, CICS supplies it for you automatically when constructing the request. If you include escaped characters in the query string, CICS passes them to the server in their escaped format.

QUERYSTRLEN*(data-value)*
Specifies the length of the query string supplied on the QUERYSTRING option, as a fullword binary value.

SESSTOKEN*(data-value)*
Specifies the session token, an 8-byte binary value that uniquely identifies this connection between CICS and a server. This value is returned by a WEB OPEN command for CICS as an HTTP client. Session tokens explains the use of the session token.

URIMAP*(data-value)*
Specifies the name (up to 8 characters, in mixed case) of a URIMAP definition that provides the path information for the specific resource within the server that the application needs to access. The URIMAP definition must be for CICS as an HTTP client (with USAGE(CLIENT) specified). Its HOST attribute must be the same as the HOST attribute of the URIMAP definition that was specified on the WEB OPEN command for this connection, or the same as the host name specified in the HOST option on the WEB OPEN command for this connection. A URIMAP definition specified on the WEB CONVERSE command applies only to this request.

If the URIMAP option is specified, do not specify the PATH or PATHLENGTH options.

USERNAME*(data-value)*
Specifies the user ID or logon name that is allowed access to this data. If the USERNAME is specified, you also need to use the PASSWORD option.

USERNAMELEN*(data-value)*
Specifies the buffer length supplied for the USERNAME option as a fullword binary variable.

Options for receiving the server's response

BODYCHARSET*(data-area)*
Specifies the character set of the HTTP response body.

The name of the character set can consist of up to 40 alphanumeric characters, including appropriate punctuation.
If the HTTP body is received into an application buffer, the character set returned is as follows:

- If the INTO or SET option is specified, and the HTTP body is converted, CICS returns the character set of the HTTP body before conversion.
- If the INTO or SET option is specified, and the HTTP body is not converted, CICS returns the character set specified in the Content-Type header. If character set information is not available, blanks are returned.

If the HTTP body is received into a named container, the character set returned is as follows:

- If the container is a CHAR container, CICS returns the character set of the encoded data.
- If the container is a BIT container, CICS returns blanks.

If the value returned is more than 40 bytes, the data is truncated. If the value returned is less than 40 bytes, the data is padded to the right with blanks.

INTO*(data-area)*
Specifies the buffer that is to contain the data being received.

MAXLENGTH*(data-value)*
Specifies the maximum amount, as a fullword binary value, of data that CICS is to pass to the application. The MAXLENGTH option applies whether the INTO or the SET option is specified for receiving data. If the data has been sent using chunked transfer-coding, CICS assembles the chunks into a single message before passing it to the application, so the MAXLENGTH option applies to the total length of the chunked message, rather than to each individual chunk. The data is measured after any code page conversion has taken place.

If the length of data exceeds the value specified and the NOTRUNCATE option is not specified, the data is truncated to that value, and the remainder of the data is discarded.

If the length of data exceeds the value specified and the NOTRUNCATE option is specified, CICS retains the remaining data and can use it to satisfy subsequent RECEIVE commands.

MEDIATYPE*(data-area)*
Specifies a 56-character data area to receive the media type (that is, the type of data content) for the body, for example `text/xml`. See IANA media types and character sets for more information about media types.

The MEDIATYPE option is used for both the sending and receiving functions of the WEB CONVERSE command. If it is specified with a value, the value is

used to construct the Content-Type header in the request, and the same field is used to receive the media type of the response that is returned by the server. If it is specified without a value, it is used only to receive the media type of the response.

NOTRUNCATE
Specifies that when the data available exceeds the length requested on the MAXLENGTH option, the remaining data is not to be discarded immediately but is to be retained for retrieval by subsequent RECEIVE commands. (If no further RECEIVE commands are issued, the data is discarded during transaction termination.)

A single RECEIVE command using the SET option and without the MAXLENGTH option receives all the remaining data, whatever its length. Alternatively, you can use a series of RECEIVE commands with the NOTRUNCATE option to receive the remaining data in appropriate chunks. Keep issuing the RECEIVE command until you are no longer getting a LENGERR response. Bear in mind that if you receive less than the length requested on the MAXLENGTH option, this does not necessarily indicate the end of the data; this could happen if CICS needs to avoid returning a partial character at the end of the data.

SET*(ptr-ref)*
Specifies a pointer reference that is to be set to the address of data received. The pointer reference is valid until the next RECEIVE command or the end of task.

STATUSCODE*(data-area)*
Specifies a data area to receive the HTTP status code sent by the server. The code is a binary halfword value. Examples are 200 (normal) or 404 (not found). Receiving the status code is optional, but you should always receive and check the status code in the following circumstances:

- If you intend to make an identical request to the server, now or during a future connection.
- If you intend to make further requests to the server using this connection.
- If your application is carrying out any further processing that depends on the information you receive in the response.

HTTP status code reference for CICS web support has basic guidance on appropriate actions for an application to take in response to the status codes for HTTP/1.1.

STATUSLEN*(data-value)*
Specifies, as a fullword binary value, the length of the data area to receive any text returned by the server to describe the status code (the STATUSTEXT option). The text is known as a reason phrase. Most reason phrases recommended for HTTP are short, but a data-area length of 256 characters is suggested here, in case the server replaces the recommended reason phrase with more detailed information.

STATUSTEXT*(data-area)*
Specifies a data area to receive any text returned by the server to describe the status code. The text is known as a reason phrase. Examples are "OK" (accompanying a 200 status code), or "Bad Request" (accompanying a 400 status code). The STATUSLEN option gives the length allowed for the text.

TOCHANNEL*(data-value)*
Specifies the name of the channel that the container belongs to. The name of the channel can consist of up to 16 alphanumeric characters, including

appropriate punctuation. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blanks are not permitted. If the name is less than 16 characters, it is padded with trailing blanks. If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

If you plan to ship your channels between CICS regions, bear in mind that you should restrict your characters to standard alphanumeric characters (A-Z 0-9 & : = , ; <>. - _) to ensure they are represented in the same way by all EBCDIC code pages.

You can specify the channel name `DFHTRANSACTION` to use a transaction channel. A transaction channel does not go out of scope when the link level changes: it is always accessible in the transaction. For more information, see Channels and containers.

If the TOCHANNEL option is not specified, then CICS assumes the current channel.

TOCONTAINER*(data-value)*
Specifies the name of the container where the data is placed. The name of the container can consist of up to 16 alphanumeric characters, including appropriate punctuation. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blanks are not permitted. If the name is less than 16 characters, it is padded with trailing blanks.

If you plan to ship your containers between CICS regions, bear in mind that you should restrict your characters to standard alphanumeric characters (A-Z 0-9 & : = , ; <>. - _) to ensure they are represented in the same way by all EBCDIC code pages.

Do not use container names starting with "DFH", unless requested to do so by CICS.

TOLENGTH*(data-area)*
Specifies a fullword binary variable that is set to the amount of data that CICS has returned to the application. Note that this might be slightly less than the limit that you set using the MAXLENGTH option, especially if a double-byte or multi-byte character set is involved, because CICS does not return a partial character at the end of the data.

- If the NOTRUNCATE option is not specified, any further data present in the message is discarded. A LENGERR response with a RESP2 value of 57 is returned if further data was present.
- If the NOTRUNCATE option is specified, any additional data is retained. A LENGERR response with a RESP2 value of 36 is returned if additional data is available. The description for the NOTRUNCATE option tells you what to do in this case.

This option is the equivalent of the LENGTH option on the WEB RECEIVE command.

If you are using the TOCONTAINER option, do not specify the TOLENGTH option.

Options for converting items sent and received

CLIENTCONV*(cvda)*
Specifies whether or not CICS translates the entity body of the HTTP request before sending, and translates the entity body of the server's response. The

default is that the entity body is converted both when the request is sent out, and when the response is received (CLICONVERT).

If you are receiving data into a named container (specified by the TOCONTAINER option), conversion does not take place.

- For the request body, you can use the CHARACTERSET option on this command to specify a character set that is suitable for the server. If conversion is requested (or happens as the default) but you do not specify a character set, the default is that CICS converts the entity body to the ISO-8859-1 character set.
- For the response body, you do not need to specify the character set used by the server. CICS identifies this by examining the Content-Type header of the message. If the header does not provide this information, or if the named character set is not supported by CICS for code page conversion, the ISO-8859-1 character set is used.
- For the application's code page, the default code page for the local CICS region (as specified in the LOCALCCSID system initialization parameter) is used, or an alternative EBCDIC code page that you specified on the WEB OPEN COMMAND.

CVDA values are as follows:

CLICONVERT
CICS converts the entity body of the request into the character set that you identify for the server, and converts the entity body of the response into a code page suitable for the application.

If the TOCONTAINER option is specified, conversion does not take place when the HTTP response is received. Instead, the media type of the HTTP response header determines whether the HTTP body is stored in a BIT or CHAR container. If the media type is a text media type, the body is stored in a CHAR container. If the media type is a non-text media type, the body is stored in a BIT container. If the HTTP response does not contain media type information, the default of text media type is assumed.

NOINCONVERT
CICS converts the entity body of the request into the character set that you identify for the server. However, CICS does not convert the entity body of the response, and it is passed to the application in the character set used by the server.

If the TOCONTAINER option is specified, conversion does not take place when the HTTP response is received. Instead, the media type of the HTTP response header determines whether the HTTP body is stored in a BIT or CHAR container. If the media type is a text media type, the body is stored in a CHAR container. If the media type is a non-text media type, the body is stored in a BIT container. If the HTTP response does not contain media type information, the default of text media type is assumed.

NOOUTCONVERT
CICS does not convert the entity body of the request, and it is sent to the server in the code page used by the application. However, CICS does convert the entity body of the response into a code page suitable for the application.

If the TOCONTAINER option is specified, conversion does not take place when the HTTP response is received. Instead, the media type of

the HTTP response header determines whether the HTTP body is stored in a BIT or CHAR container. If the media type is a text media type, the body is stored in a CHAR container. If the media type is a non-text media type, the body is stored in a BIT container. If the HTTP response does not contain media type information, the default of text media type is assumed.

NOCLICONVERT
CICS does not convert the entity body of the request, and it is sent to the server in the code page used by the application. CICS does not convert the entity body of the response, and it is passed to the application in the character set used by the server.

If the TOCONTAINER option is specified, conversion does not take place when the HTTP response is received. Instead, the media type of the HTTP response header determines whether the HTTP body is stored in a BIT or CHAR container. If the media type is a text media type, the body is stored in a CHAR container. If the media type is a non-text media type, the body is stored in a BIT container. If the HTTP response does not contain media type information, the default of text media type is assumed.

Conditions

122 CHANNELERR
RESP2 values are:

1 The name specified by the TOCHANNEL option contains an illegal character or combination of characters.

2 The channel specified on the CHANNEL option could not be found.

110 CONTAINERERR
RESP2 values are:

1 The name specified by the TOCONTAINER option contains an illegal character or combination of characters.

2 The container specified by the CONTAINER option could not be found.

19 NOTOPEN
RESP2 values are:

27 Invalid session token.

16 INVREQ
RESP2 values are:

10 Invalid response header.

11 Action code invalid.

13 Close status invalid.

15 Code page conversion failure.

17 Expect-100 request was rejected by the server.

22 Invalid chunk size.

32 Media type invalid.

33 Method does not support a body.

34 Method requires a body.

41 The connection has been closed.

43 The DOCSTATUS value specified is invalid.

45 The character set specified is invalid.

46 The CLIENTCONV option is invalid.

49 The format of the path option is invalid.

54 The HTTP method is not valid.

63 URIMAP object disabled.

64 Host in URIMAP definition does not match host specified when this session was opened.

67 The content of the response does not conform to HTTP format. The error is generated because there is a syntax problem.

74 The connection has been closed (CICS sent a Connection: close header to the server, or the server may have timed out due to inactivity on this connection).

76 MEDIATYPE option required.

79 Pipelining is in progress. WEB CONVERSE command cannot be used.

80 CHARACTERSET cannot be specified with NOSRVCONVERT.

142 AUTHENTICATE is invalid. The CVDA is not NONE or BASICAUTH.

144 One or more of the Web command parameters is invalid.

145 Either CHANNEL was not specified with CONTAINER, or TOCHANNEL was not specified with TOCONTAINER (and there is no current channel).

146 The named container is a read-only container.

147 Internal conversion error.

150 Conversion requested, but data to be sent is in a DATATYPE BIT container.

22 LENGERR

RESP2 values are:

5 The PATHLENGTH option value was not greater than zero.

8 The QUERYSTRLEN option value was not greater than zero.

16 Invalid MAXLENGTH.

36 Partial response body returned. Use additional RECEIVEs to obtain remainder.

50 The FROMLENGTH option value was not greater than zero.

57 The response body exceeds the length specified, and the remainder of the body has been discarded.

58 The status text exceeds the length specified.

59 The STATUSLEN option value was not greater than zero.

139 USERNAMELEN is not positive.

140 PASSWORDLEN is not positive.

13 NOTFND
RESP2 values are:

61 The URIMAP object specified was not found.

112 TOKENERR
RESP2 values are:

47 The document token specified is invalid or the document has been deleted.

17 IOERR
RESP2 values are:

42 Socket error.

124 TIMEDOUT
RESP2 values are:

62 Timeout on socket receive.

156 Timeout on socket send.

70 NOTAUTH
RESP2 values are:

100 Path barred by security exit.

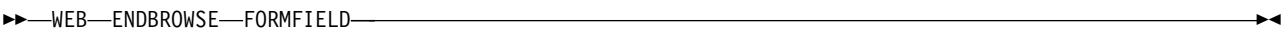
110 XWBAUTH error. The XWBAUTH global user exit has issued a UERCERR return code because the XWBAUTH exit is required but cannot return a valid response.

This error code is issued when the following are true: BASICAUTH is specified; USERNAME, PASSWORD, or both are omitted; XWBAUTH is inactive or returns a response of UERCERR.

WEB ENDBROWSE FORMFIELD

Signal the end of a formfield browse in an HTML form.

WEB ENDBROWSE FORMFIELD



Conditions: INVREQ

This command is threadsafe.

Description

WEB ENDBROWSE FORMFIELD terminates the browse of a set of name-value pairs in an HTML form. The form is part of the body of an HTTP request being processed by the current CICS task. No information is returned on the ENDBROWSE.

Conditions

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

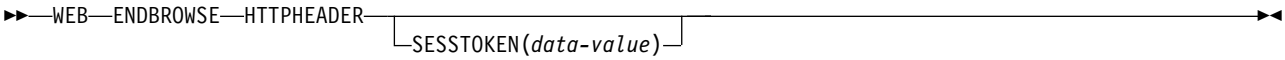
3 The command is being issued for a non-HTTP request.

4 The command is being issued before a WEB STARTBROWSE command is issued.

WEB ENDBROWSE HTTPHEADER

Signal the end of an HTTP header browse.

WEB ENDBROWSE HTTPHEADER



Conditions: INVREQ, NOTOPEN

This command is threadsafe.

Description

WEB ENDBROWSE HTTPHEADER terminates the browse. No information is returned on the ENDBROWSE. The SESSTOKEN option is required if the HTTP header information is part of a response sent to CICS as an HTTP client.

Options

SESSTOKEN*(data-value)*
For CICS as an HTTP client, this option is required. It specifies the session token, an 8-byte binary value that uniquely identifies a connection between CICS and a server. This value is returned by a WEB OPEN command for CICS as an HTTP client. Session tokens explains the use of the session token.

Conditions

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request.

4 The command is being issued before a WEB STARTBROWSE command is issued.

19 NOTOPEN
RESP2 value is:

27 Invalid session token.

WEB ENDBROWSE QUERYPARM

Finish browsing query string data in a URL.

WEB ENDBROWSE QUERYPARM



Conditions: INVREQ

This command is threadsafe.

Description

WEB ENDBROWSE QUERYPARM terminates the browse of a set of keyword parameters, consisting of name and value pairs, from a query string in a URL. No information is returned on the ENDBROWSE.

Conditions

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

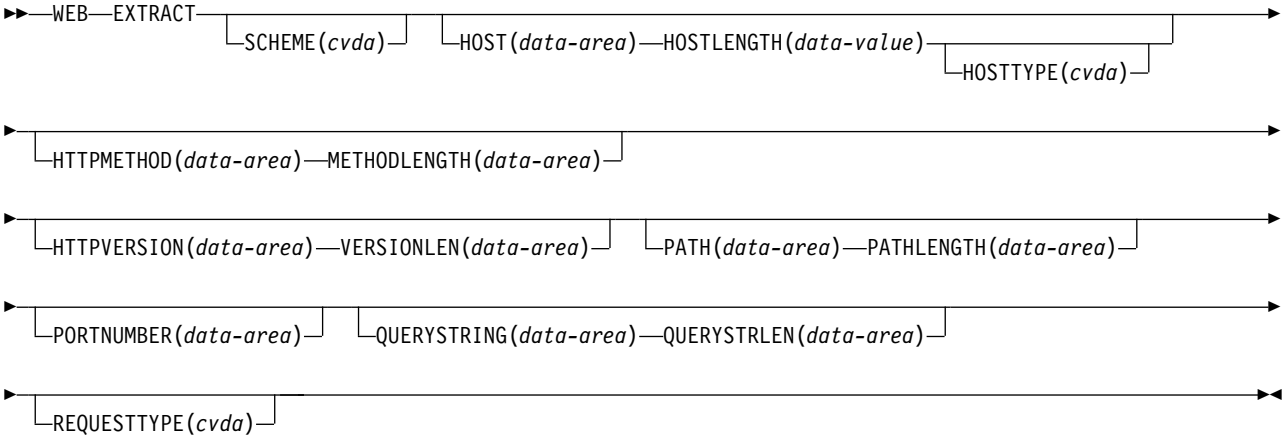
3 The command is being issued for a non-HTTP request.

4 The command is being issued before a WEB STARTBROWSE command is issued.

WEB EXTRACT

Obtain information about an HTTP request that has been made to CICS as an HTTP server or about a connection between an Internet server and CICS as an HTTP client. This command is a synonym of EXTRACT WEB.

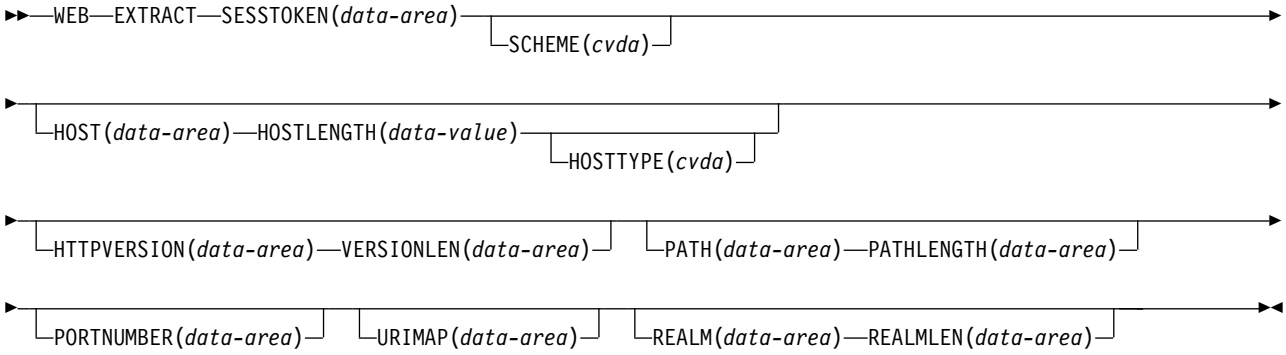
WEB EXTRACT (CICS as an HTTP server)



Conditions: INVREQ, IOERR, LENGERR, NOTFND, NOTOPEN, TIMEDOUT

This command is threadsafe.

WEB EXTRACT (CICS as an HTTP client)



Conditions: INVREQ, IOERR, LENGERR, NOTFND, NOTOPEN, TIMEDOUT

This command is threadsafe.

Description

For CICS as an HTTP server, WEB EXTRACT enables an application to obtain information about the most recent HTTP request that has been made to CICS by a Web client and assigned to the application for handling.

For CICS as an HTTP client, when the SESSTOKEN option is specified, the command enables an application to obtain information about a connection that it has opened with a server. The information returned to the application comprises global information about the connection, such as the host name of the server and its HTTP version. Information about specific requests made by the application, and responses made by the server, is not available using this command. The WEB RECEIVE command is used to receive information from a server response.

Options

HOST*(data-area)*

For CICS as an HTTP server, HOST specifies a buffer to contain the host component of the URL, as specified either in the Host header field for the request or in the request line (if an absolute URI was used for the request). The port number is presented separately using the PORTNUMBER option.

For CICS as an HTTP client, with the SESSTOKEN option, HOST specifies a buffer to contain the host name of the server in the connection identified by the SESSTOKEN option. The port number is presented separately using the PORTNUMBER option.

An IPv4 or IPv6 address can represent the host name. IPv4 addresses are returned as native IPv4 dotted decimal addresses; for example, `1.2.3.4`. IPv6 addresses are returned as native IPv6 colon hexadecimal addresses; for example, `::a:b:c:d`

For information on IP addresses, see IP addresses.

HOSTLENGTH*(data-area)*

Specifies the length of the buffer supplied on the HOST option, as a fullword binary variable, and is set to the length of the data returned to the application. 116 characters is an appropriate size to specify for this data area. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

HOSTTYPE*(cvda)*
Returns the address format of the HOST option. CVDA values are as follows:

HOSTNAME
The HOST option contains a character host name. The IP address that corresponds to the host name is looked up in the domain name server.

IPV4 The address is a dotted decimal IPv4 address.

IPV6 The address is a colon hexadecimal IPv6 address.

NOTAPPLIC
An incorrect host address was returned (`HOST=0.0.0.0`).

HTTPMETHOD*(data-area)*
For CICS as an HTTP server, this option specifies a buffer to contain the HTTP method string on the request line of the message.

This option is not relevant for CICS as an HTTP client.

HTTPVERSION*(data-area)*
For CICS as an HTTP server, this option specifies a buffer to contain the HTTP version for the Web client, as stated on its request.

For CICS as an HTTP client (with the SESSTOKEN option), this option specifies a buffer to contain the HTTP version of the server in the connection identified by the SESSTOKEN option. If CICS does not already know the HTTP version of the server, CICS makes a request to the server with the OPTIONS method to find out this information.

`1.1` indicates HTTP/1.1, and `1.0` indicates HTTP/1.0 or lower.

METHODLENGTH*(data-area)*
Specifies the length of the buffer supplied on the HTTPMETHOD option, as a fullword binary variable, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

PATH*(data-area)*
For CICS as an HTTP server, this option specifies a buffer to contain the path specified in the request line of the message.

For CICS as an HTTP client (with the SESSTOKEN option), this option specifies a buffer to contain the default path that applies to requests made using the connection. If a URIMAP definition was specified on the WEB OPEN command for the connection, the default path is the path specified in the URIMAP definition. Otherwise, the default path is a single forward slash.

PATHLENGTH*(data-area)*
Specifies the length of the buffer supplied on the PATH option, as a fullword binary variable, and is set to the length of the data returned to the application. 256 characters is an appropriate size to specify for this data-area. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

PORTNUMBER*(data-area)*
For CICS as an HTTP server, this option returns a data area containing the port number specified in the request line of the message.

For CICS as an HTTP client (with the SESSTOKEN option), this option returns a data containing the port number used to access the server in the connection specified by the SESSTOKEN option.

The value returned in the data area is a fullword binary value.

Well-known port numbers for a service are normally omitted from the URL. If the port number is not present in the URL, the command identifies and returns it based on the scheme. For HTTP, the well-known port number is 80, and, for HTTPS, the well-known port number is 443. If a port number is returned that is not the default for the scheme, you must specify the port number explicitly to gain access to the URL; for example, if you are using this information in a WEB OPEN command.

QUERYSTRING*(data-area)*
For CICS as an HTTP server, this option specifies a buffer to contain the query string on the request line of the message. The query string is the value or values encoded after the question mark (?) delimiting the end of the path. The query string is returned in its escaped form.

This option is not relevant for CICS as an HTTP client.

QUERYSTRLEN*(data-area)*
Specifies the length of the buffer supplied on the QUERY option, as a fullword binary variable, and is set to the length of the data returned to the application (the query string). 256 characters is an appropriate size to specify for this data area. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

REALM*(data-area)*
Specifies, for CICS as an HTTP client, the realm or security environment that contains the data that you are requesting. If you are issuing a command in response to an HTTP 401 message, REALM is the realm value in the most recently received WWW-Authenticate header.

REALMLEN*(data-area)*
Specifies, for CICS as an HTTP client, the buffer length supplied for the REALM option, as a fullword binary variable. If you are issuing a command in response to an HTTP 401 message, REALMLEN is the length of the realm name in the most recently received WWW-Authenticate header.

REQUESTTYPE*(cvda)*
For CICS as an HTTP server, this option specifies the type of request received. This option is not relevant for CICS as an HTTP client. CVDA values are as follows:

HTTPYES
Indicates an HTTP request.

HTTPNO
Indicates a non-HTTP request.

SCHEME*(cvda)*
For both CICS as an HTTP server, and CICS as an HTTP client (with the SESSTOKEN option), this option returns the scheme used for the connection between CICS and the Web client or server. CVDA values are as follows:

HTTP Is the HTTP protocol, without SSL.

HTTPS
Is the HTTPS protocol, which is HTTP with SSL.

SESSTOKEN*(data-value)*
For CICS as an HTTP client, this option is required. It specifies the session token, an 8-byte binary value that uniquely identifies a connection between CICS and a server. This value is returned by a WEB OPEN command for CICS as an HTTP client. Session tokens explains the use of the session token. For the command, information is returned about the specified connection.

This option is not relevant for CICS as an HTTP server.

URIMAP*(data-area)*
For CICS as an HTTP client (with the SESSTOKEN option), this option returns the 8-character name (in mixed case) of any URIMAP definition that was specified on the WEB OPEN command to open the connection specified by the SESSTOKEN option. The INQUIRE URIMAP command can be used to find information about the attributes of this URIMAP definition.

This option is not relevant for CICS as an HTTP server.

VERSIONLEN*(data-area)*
Specifies the length of the buffer supplied on the HTTPVERSION option, as a fullword binary variable, and is set to the length of the data returned to the application.

Conditions

16 INVREQ
RESP2 values:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request. This command is set only if one or more of HTTPMETHOD, HTTPVERSION, or PATH is specified and the request is a non-HTTP request).

41 The connection has closed. This is a WEB EXTRACT (Client) error only. The server may have timed out due to inactivity on this connection.

67 The content of the response does not conform to HTTP format. The error is generated because there is a syntax problem. This error is for WEB EXTRACT (Client) only.

71 A chunked transfer-coding error has occurred. This error is for WEB EXTRACT (Client) only.

144 One or more of the Web command parameters is invalid. This error is for WEB EXTRACT (Client) only.

17 IOERR
RESP2 values:

42 Socket error.

22 LENGERR
RESP2 values:

4 The method exceeds the length specified (METHODLENGTH option).

5 The PATHLENGTH option value was not greater than zero.

6 The HTTP version exceeds the length specified (VERSIONLEN option).

7 The VERSIONLEN option value was not greater than zero.

8 The query string exceeds the length specified (QUERYSTRLEN option).

21 The HOSTLENGTH option value was not greater than zero.

29 The host name exceeds the length specified (HOSTLENGTH option).

30 The path exceeds the length specified (PATHLENGTH option).

141 REALMLEN is not positive, or is not large enough to contain the realm value returned in the HTTP 401 response.

13 NOTFND
RESP2 values:

155 Request line information not found.

19 NOTOPEN
RESP2 values:

27 Session token not valid.

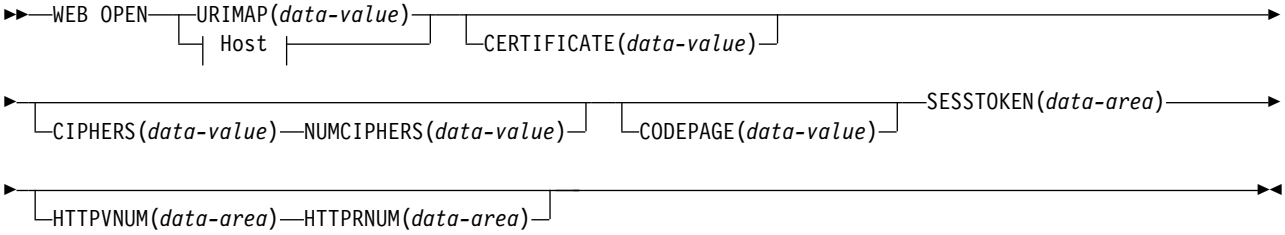
124 TIMEDOUT
RESP2 values:

62 Timeout on socket receive.

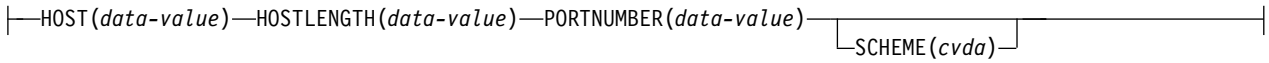
WEB OPEN

Open a connection to a server for CICS as an HTTP client.

WEB OPEN



Host:



Conditions: IOERR, INVREQ, LENGERR, NOTFND, NOTAUTH, TIMEDOUT

This command is threadsafe.

Description

WEB OPEN enables an application program, through CICS Web support, to open a connection with a specified host on an HTTP server on the Internet. When the connection is open, the application program can make HTTP client requests to the server and receive responses from it.

When you open the connection, you can specify a URIMAP resource that contains the information about the URL for the connection. You can specify this information directly on the WEB OPEN command instead of using a URIMAP resource. However, using a URIMAP resource has the following advantages:

- System administrators can manage any changes to the endpoint of the connection, so you do not need to recompile your applications if the URL for your request changes.
- If you are using SSL, you can specify an SSL client certificate or cipher suite codes in the URIMAP resource, so that system administrators can manage any changes to these certificates and codes.

- You can choose to make CICS keep the connections that were opened with the URIMAP resource open after use, and place them in a pool for reuse by another application or another instance of the same application. Connection pooling is only available when you specify a URIMAP resource that has the SOCKETCLOSE attribute set. For more information about the performance benefits of connection pooling, see Connection pooling for HTTP client performance.

For information about creating a URIMAP resource for a client request, see Creating a URIMAP resource for CICS as a HTTP client.

The WEB OPEN command drives the XWBOPEN user exit, which can make the connection to the server go through a proxy server, if required.

Note: If the connection request does not complete within the deadlock timeout interval (specified in the DTIMOUT attribute of the TRANSACTION definition for the transaction that starts the user application), CICS returns a TIMEDOUT response to the application. Setting DTIMOUT to NO, or allowing it to default to NO, means that the application is prepared to wait indefinitely.

Options

CERTIFICATE*(data-value)*
Specifies the label of the X.509 certificate that is to be used as the SSL client certificate during the SSL handshake. Certificate labels can consist of up to 32 alphanumeric characters. This option is relevant only when the HTTPS option is specified. If HTTPS is specified, but the CERTIFICATE option is omitted, the default certificate defined in the key ring for the CICS region user ID is used. The certificate must be stored in a key ring in the external security manager's database. For more information, see Building a key ring manually.

CIPHERS*(data-value)*
Specifies a string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes. The cipher suite codes are used when SSL is active for the connection, so this option is relevant only when the HTTPS option is specified. They indicate the method of encryption to be used for this connection.

Use the NUMCIPHERS option to specify the number of cipher suite codes in your list. The codes that are available depend on what level of encryption has been specified by the ENCRYPTION system initialization parameter. If you specify any cipher codes that are not in the default list for the active encryption level, they are ignored. For more information on cipher suites, see Cipher suites and cipher suite specification files.

You can specify the URIMAP option to use this information directly from an existing URIMAP definition, in which case the CIPHERS option is not required. You can still specify the CIPHERS option, and the setting in the URIMAP definition is overridden by any codes that you specify for this option.

Note: **EXEC CICS WEB OPEN** does not support the use of an SSL cipher suite specification file in the CIPHERS option.

If you omit the CIPHERS option and the URIMAP option, but SSL is active for the connection, the default cipher list for the encryption level for the running system is used.

CODEPAGE*(data-value)*
Specifies a code page that is suitable for the application program. The code

page name can be up to 8 alphanumeric characters. The default is the default code page for the local CICS region, as specified in the LOCALCCSID system initialization parameter. The code page applies for the duration of this connection. When the server returns a response to an HTTP request, if conversion is requested (which is the default), CICS converts the request body into this code page before passing it to the application.

The standard CICS form of this code page name consists of the code page number (or more generally CCSID) written using 3 to 5 decimal digits as necessary then padded with trailing spaces to 8 characters. For code page 37, which is fewer than 3 digits, the standard form is 037. CICS also accepts any decimal number of up to 8 digits (padded with trailing spaces) in the range 1 to 65535 as a code page name, even if it is not in the standard form.

HOST*(data-value)*
Specifies the host name on the server to which you want to connect. You can extract this information from a known URL using the WEB PARSE URL command, or from an existing URIMAP definition using the WEB EXTRACT URIMAP command. You can specify the URIMAP option to use this information directly from an existing URIMAP definition, in which case the HOST option is not required. Client HTTP connections can only be pooled for reuse when you specify the URIMAP option; using the HOST option does not enable connection pooling, even if you extract the information from a URIMAP definition.

A character host name, IPv4 address, or IPv6 address can represent the host name. If you specify an IPv6 address (or a host name that resolves to an IPv6 address), ensure that you are operating in a dual-mode (IPv4 and IPv6) environment and that the client or server that you are communicating with is also operating in a dual-mode (IPv4 and IPv6) environment.

For more information on IPv6, see Understanding IPv6 and CICS.

You can specify IPv4 and IPv6 addresses in a number of formats. For information on IP addresses, see IP addresses.

If you require a port number, you must not include the port number as part of the HOST option. Use the PORTNUMBER option instead.

HOSTLENGTH*(data-value)*
Specifies as a fullword binary value, the length of the host name. This information is returned if you use the WEB PARSE URL command to parse a URL, or it can be extracted from an existing URIMAP definition using the WEB EXTRACT URIMAP command. You can specify the URIMAP option to use this information directly from an existing URIMAP definition, in which case the HOSTLENGTH option is not required.

HTTPRNUM*(data-area)*
Returns the release number for the HTTP version of the server, as a halfword binary value. (HTTPVNUM returns the version number.) For example, if the server is at HTTP/1.0 level, HTTPRNUM returns 0.

HTTPVNUM*(data-area)*
Returns the version number for the HTTP version of the server, as a halfword binary value. (HTTPRNUM returns the release number.) For example, if the server is at HTTP/1.0 level, HTTPVNUM returns 1.

If you specify the HTTPVNUM and HTTPRNUM options, CICS obtains the HTTP version information when it opens the connection to the server. If the server does not provide HTTP version information in response to this request, or the version is lower than 1.0, CICS assumes that it is at HTTP/1.0 level.

Specify these options if it is essential for you to check the HTTP version information to confirm whether a planned action by your application, before or during its first request, will succeed. Actions dependent on the HTTP version include:

- Writing HTTP headers that request an action which might not be carried out correctly by a server below HTTP/1.1 level
- Using HTTP methods that might be unsuitable for servers below HTTP/1.1 level
- Using chunked transfer-coding
- Sending a pipelined sequence of requests

The additional HTTP request made by CICS to obtain the HTTP version information has an impact on performance, so do not specify these options if it is not necessary at this stage. When the first response has been received from the server, you can obtain this information using the WEB EXTRACT command.

NUMCIPHERS*(data-value)*
Specifies, as a halfword binary value, the number of cipher suite codes that you specified for the CIPHERS option.

PORTNUMBER*(data-value)*
Specifies the port number, as a fullword binary value. You specify the port number only if it is *not* the default for the specified scheme. For HTTP, the default port number is 80, and for HTTPS, the default port number is 443. Port number information can be extracted from a known URL using the WEB PARSE URL command, or from an existing URIMAP definition using the WEB EXTRACT URIMAP command. You can specify the URIMAP option to use this information directly from an existing URIMAP definition, in which case the PORTNUMBER option is not required. Client HTTP connections can only be pooled for reuse when you specify the URIMAP option; using the PORTNUMBER option does not enable connection pooling, even if you extract the information from a URIMAP definition.

SCHEME*(cvda)*
Specifies the scheme that is to be used for the connection to the server, which can be with or without SSL. CVDA values are as follows:

HTTP The HTTP protocol, without SSL.

HTTPS

The HTTPS protocol, which is HTTP with SSL. If HTTPS is used, the CICS address space must be enabled for SSL.

This information can be extracted from a known URL using the WEB PARSE URL command, or from an existing URIMAP definition using the WEB EXTRACT URIMAP command. You can specify the URIMAP option to use this information directly from an existing URIMAP definition, in which case the SCHEME option is not required. Client HTTP connections can be pooled for reuse only when you specify the URIMAP option; using the SCHEME option does not enable connection pooling, even if you extract the information from a URIMAP definition.

SESSTOKEN*(data-area)*
Returns the session token, an 8-byte binary value that uniquely identifies this application's use of the connection between CICS and the server. The session token must be used on all CICS WEB commands that relate to this connection. See Session tokens for information about use of the session token.

URIMAP*(data-value)*

Specifies the name, up to 8 characters in mixed case, of a URIMAP definition that provides the following information:

- The scheme that is to be used for the connection to the server.
- The host name on the server to which you want to connect.
- A port number, if required.
- A path component for the URI, representing the resource on the server that you want to access. This path becomes the default path for WEB SEND or WEB CONVERSE commands relating to this connection, but it can be overridden by specifying another path on the WEB SEND or WEB CONVERSE command.
- The expiry period for pooled connections that are opened using the URIMAP resource. Connection pooling is enabled when you specify an expiry period in the URIMAP definition using the SOCKETCLOSE attribute, and name the URIMAP resource on the WEB OPEN command.
- The label of the X.509 certificate that is to be used as the SSL client certificate, if required.
- The cipher suite codes that can be used for the connection.

If the URIMAP option is specified, do not specify the CERTIFICATE, HOST, HOSTLENGTH, PORTNUMBER, PORTLENGTH, or SCHEME options. The CIPHERS and NUMCIPHERS options can be omitted or specified in the command; if specified, they override these settings in the URIMAP definition.

Note: **EXEC CICS WEB OPEN** does not support the use of an SSL cipher suite specification file in the CIPHERS option.
The URIMAP definition must be for CICS as an HTTP client, with USAGE(CLIENT) specified.

Conditions

17 IOERR

RESP2 values:

38 Proxy error.

42 Socket error. One possible cause of this response is that the system initialization parameter **TCPIP** is set to NO, or defaulted to NO, for this CICS region.

16 INVREQ

RESP2 values:

14 Code page incorrect.

22 Incorrect chunk received during the initial HTTP request using the OPTIONS method.

23 Incorrect client certificate.

40 Scheme incorrect.

41 Server closed the connection during the initial HTTP request using the OPTIONS method.

48 The format of the host option is incorrect.

63 URIMAP object is not available.

66 An error occurred in processing for the XWBOPEN exit.

67 The content of the response does not conform to HTTP format. The error is generated because there is a syntax problem.

96 SSL not supported.

137 All requested cipher codes have been rejected.

138 The port number is greater than 65535.

144 One or more of the Web command parameters is incorrect.

22 LENGERR
RESP2 values:

21 Incorrect host length.

13 NOTFND
RESP2 values:

20 Host name is not resolved by name server or the format of the host option is incorrect.

39 Unknown proxy.

61 The URIMAP object specified was not found.

70 NOTAUTH
RESP2 values:

100 Host name barred by security exit.

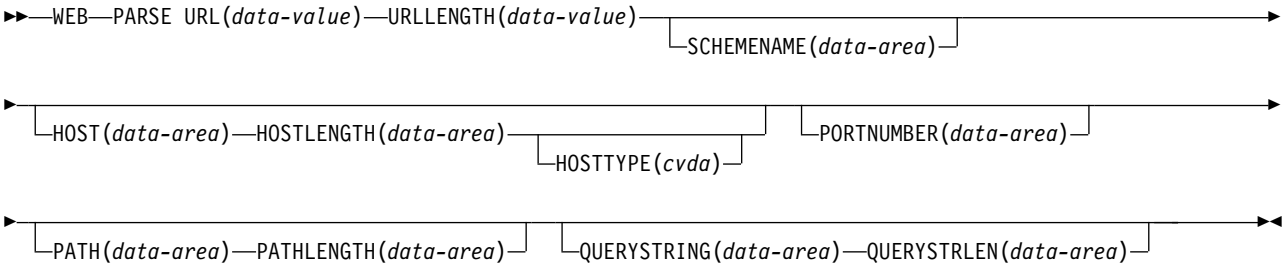
124 TIMEDOUT
RESP2 values:

62 Timeout on socket receive.

WEB PARSE URL

Break down a URL string into its component parts.

WEB PARSE URL



Conditions: INVREQ, LENGERR

This command is threadsafe.

Description

With WEB PARSE URL, you can break down a URL string into its component parts: scheme, host, port, path, and query string. The components of a URL explains these components. You can use this process to examine the construction of

the URL and to separate the components. The returned information can be used in the WEB OPEN command to open a client connection to the host named in the URL.

Any escape sequences found in the URL are checked for validity. An escape sequence consists of the percent character (%) followed by two hexadecimal characters. Valid hexadecimal characters are the digits 0 to 9 and the letters A to F.

Note that where the string input to the WEB PARSE URL command has been delimited in the correct way for a URL, the command does not detect incorrect content, such as a host name that does not represent an existing host on the Internet, or a character that is not permitted in a URL.

Options

HOST*(data-area)*
Returns the host component of the URL. This value can be either a character host name or a numeric IP address. If a port number is specified explicitly in the URL, the port number is returned separately as the PORTNUMBER option.

An IPv4 or IPv6 address can represent the host name. IPv6 addresses are returned as native IPv6 colon hexadecimal addresses, for example, `::a:b:c:d`. If you specify an IPv6 address in a URL, for example, `http://[::a:b:c:d]:80`, HOST returns the address without brackets.

Use the characters `X'BA'` and `X'BB'` (code page 37) to represent square brackets when you specify IPv6 addresses.

For information on IP addresses, see IP addresses.

HOSTLENGTH*(data-area)*
Specifies the length of the buffer supplied on the HOST option, as a fullword binary variable, and is set to the length of the data returned to the application (the host name). 116 characters is suggested as an appropriate size to specify for this data area. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

HOSTTYPE*(cvda)*
Returns the address format of the HOST option. CVDA values are as follows:

HOSTNAME
The HOST option contains a character host name. The IP address that corresponds to the host name is looked up in the domain name server.

IPV4 The address is a dotted decimal IPv4 address.

IPV6 The address is a colon hexadecimal IPv6 address.

PATH*(data-area)*
Returns the path component of the URL.

PATHLENGTH*(data-area)*
Specifies the length of the buffer supplied on the PATH option, as a fullword binary variable, and is set to the actual length of the data returned to the application (the path component of the URL). 256 characters is suggested as an appropriate size to specify for this data area. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

PORTNUMBER*(data-area)*
Returns as a fullword binary data area, the port number that is specified in, or appropriate for, the URL. Port numbers are sometimes specified explicitly in a URL, following the host name. However, well-known port numbers for a

service are normally omitted from a URL. If the port number is not present in the URL, the WEB PARSE URL command identifies and returns it based on the scheme. For HTTP, the well-known port number is 80, and, for HTTPS, the well-known port number is 443. If a port number is returned that is not the default for the scheme, you must specify the port number explicitly to gain access to the URL, for example, if you are using this information in a WEB OPEN command.

QUERYSTRING*(data-area)*
Returns the query string from the URL. The query string is the value or values encoded after the question mark (?) delimiting the end of the path. The query string is returned in its escaped form.

QUERYSTRLEN*(data-area)*
Specifies the length of the buffer supplied on the QUERYSTRING option, as a fullword binary variable, and is set to the length of the data returned to the application (the query string). 256 characters is suggested as an appropriate size to specify for this data area. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

SCHEMENAME*(data-area)*
Returns the scheme component of the URL, as a 16-character data area. Only the HTTP and HTTPS schemes (the HTTP protocol with and without SSL) are supported by CICS and can be used in a WEB OPEN command.

The scheme name is always returned in uppercase.

URL*(data-value)*
Specifies the complete URL string.

URLLENGTH*(data-value)*
Specifies the length of the buffer containing the URL string, as a fullword binary value.

Conditions

16 INVREQ
RESP2 values:

28 Incorrect URL.

65 Bad escape sequence.

22 LENGERR
RESP2 values:

8 Length of query string returned is greater than QUERYSTRLEN.

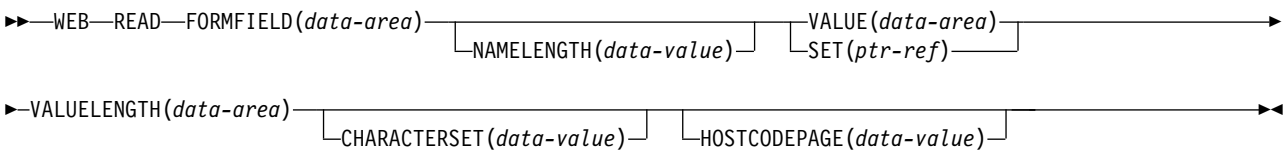
29 Length of host name returned is greater than HOSTLENGTH.

30 Length of path returned is greater than PATHLENGTH.

WEB READ FORMFIELD

Retrieve the value of a field from an HTML form.

WEB READ FORMFIELD



Conditions: INVREQ, LENGERR, NOTFND

This command is threadsafe.

Description

WEB READ FORMFIELD retrieves the value of a specific field from an HTML form. The name of the form field is given in the FORMFIELD parameter. The form data is sent as part of an HTTP request being processed by the current CICS task.

The Web client sends form data in a query string when the GET method is used, and in the entity body when the POST method is used. CICS can extract the data from either of these locations.

The form data is returned in its unescaped form (see Reserved and excluded characters for an explanation of this).

If the data that is received represents a file, the uploaded file does *not* undergo code page conversion.

CICS only reads form data when CICS is the HTTP server. The facility is not available when CICS is an HTTP client.

Options

CHARACTERSET(name)
specifies the 40–character name of the character set that is required for encoding the form data. This option should match the forms encoding determined by the corresponding HTML form (see How the client encoding is determined for more information). CICS does not support all the character sets named by IANA. HTML coded character sets lists the IANA character sets that are supported by CICS for code page conversion.

CLNTCODEPAGE(name)
This option is supported for upgrade purposes only. CHARACTERSET replaces it. The action taken by CICS is the same for both keywords.

FORMFIELD*(data-area)*
specifies the name of the form field to extract. It is a string of text containing the name of the requested field. The string of text supplied is not case sensitive.

HOSTCODEPAGE*(data-value)*
specifies the 8–character name of the CICS (host) code page required by the application program, into which the form data is to be converted. This code page is normally an EBCDIC code page.

The standard CICS form of a host code page name consists of the code page number (or more generally CCSID) written using 3 to 5 decimal digits as necessary then padded with trailing spaces to 8 characters. For code page 37,

which is fewer than 3 digits, the standard form is 037. CICS now also accepts any decimal number of up to 8 digits (padded with trailing spaces) in the range 1 to 65535 as a code page name, even if it is not in the standard form.

If the code page is not specified, the data is returned in the EBCDIC code page specified by the LOCALCCSID system initialization parameter (which applies to the local CICS region, and has a default of 037), provided that the specified code page is supported by the CICS web interface. The code page is supported if it is one of a list of EBCDIC code pages that are recognized by CICS as being sufficiently standard to allow successful parsing of the web headers (this includes all SBCS CECP and Euro code pages). Otherwise, CICS returns the data in the default EBCDIC code page 037 instead.

NAMELENGTH*(data-value)*
specifies the length, as a fullword binary value, of the form field name.

SET*(ptr-ref)*
specifies a pointer reference that is to be set to the address of data received. The pointer reference is valid until the end of the task.

VALUE*(data-area)*
specifies the buffer to contain the value of the named form field. CICS unescapes any escaped characters before placing them in the buffer.

VALUELENGTH*(data-area)*
specifies the length, as a fullword binary value, of the form field value. The actual length of the value is returned in this data area. If you specify the VALUE option, VALUELENGTH specifies the maximum length of the data that the program accepts. If the value exceeds the length of the buffer, it is truncated. If the length of the form field value is less than the size of the buffer, the form field value is placed in the leftmost byte positions.

Conditions

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request.

11 The client code page cannot be found.

12 The host code page cannot be found.

13 No forms data has been supplied in the HTTP request.

14 The code page combination for client and server is invalid.

17 Invalid forms data was found in the input message.

22 LENGERR
RESP2 values are:

1 The length in NAMELENGTH or VALUELENGTH is less than or equal to zero.

4 The form field name has been truncated during a read operation because the receiving buffer is too small.

5 The form field value has been truncated during a read operation because the receiving buffer is too small.

153 The form type is unknown.

154 A boundary string was expected in the forms data, but was not found.

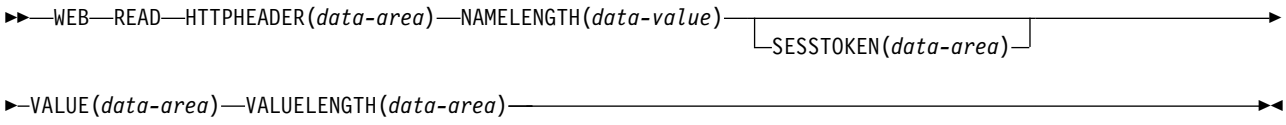
13 NOTFND
RESP2 values are:

1 The form field with the given name cannot be found.

WEB READ HTTPHEADER

Extract HTTP header information.

WEB READ HTTPHEADER



Conditions: INVREQ, LENGERR, NOTFND, NOTOPEN

This command is threadsafe.

Description

WEB READ HTTPHEADER enables an application to extract HTTP header information from a message. When CICS is an HTTP server, the message is a request from a Web client. When CICS is an HTTP client, the message is a response from a server, and the SESSTOKEN option is specified.

For CICS as an HTTP server, you can use the WEB READ HTTPHEADER command either before, or after, you use the WEB RECEIVE command to receive the body of the message. For CICS as an HTTP client, you must first receive the message using the WEB RECEIVE command, and then you can read the headers using the WEB READ HTTPHEADER command.

HTTP header reference for CICS web support lists HTTP/1.1 headers that you are likely to receive, and gives guidance for the actions that you might take in response to them.

The HTTP header browsing commands (WEB STARTBROWSE HTTPHEADER, WEB READNEXT HTTPHEADER, WEB ENDBROWSE HTTPHEADER) can be used to browse through all the HTTP header information for a message.

Options

HTTPHEADER*(data-area)*
specifies the name of the HTTP header to be extracted.

NAMELENGTH*(data-value)*
specifies the length, as a fullword binary value, of the HTTP header name.

SESSTOKEN*(data-value)*
For CICS as an HTTP client, this option is required. It specifies the session token, an 8-byte binary value that uniquely identifies a connection between CICS and a server. This value is returned by a WEB OPEN command for CICS as an HTTP client. Session tokens explains the use of the session token.

VALUE*(data-area)*
specifies the buffer to contain the value of the HTTP header being extracted.

VALUELENGTH*(data-area)*
specifies the length of the buffer supplied on the VALUE option, as a fullword binary variable, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

Conditions

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request.

43 No HTTP headers found.

22 LENGERR
RESP2 values are:

1 The length in VALUELENGTH is not greater than zero (CICS as an HTTP server).

2 The header value has been truncated because the receiving buffer is too small (CICS as an HTTP server).

35 The length in NAMELENGTH is not greater than zero (CICS as an HTTP client).

52 The header value has been truncated because the receiving buffer is too small (CICS as an HTTP client).

55 The length in VALUELENGTH is not greater than zero (CICS as an HTTP client).

13 NOTFND
RESP2 value is:

1 The header with the given name could not be found.

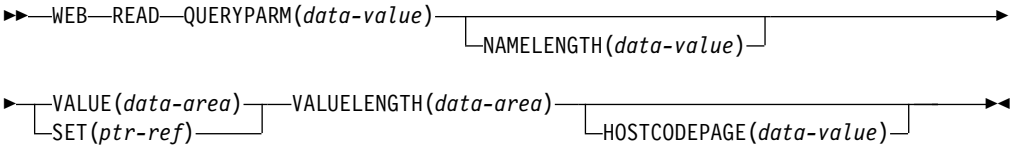
19 NOTOPEN
RESP2 value is:

27 Invalid session token.

WEB READ QUERYPARM

Read name and value pairs from a query string in a URL.

WEB READ QUERYPARM



Conditions: INVREQ, LENGERR, NOTFND

This command is threadsafe.

Description

The WEB READ QUERYPARM command reads a keyword parameter, consisting of a name and value pair, from a query string in a URL, and returns it in a specified code page. Escaped characters in the query string data are unescaped into the code page.

The WEB READ QUERYPARM command processes query string data for all HTTP methods, including GET, POST, PUT, and DELETE. You can continue to use the existing WEB READ FORMFIELD command for forms (messages with the media types application/x-www-form-urlencoded or multipart/form-data). Although the WEB READ FORMFIELD command can read name and value pairs from a query string, it does so only when the HTTP method is GET, because it assumes that the message is an HTML form.

You can use the query string browsing commands (WEB STARTBROWSE QUERYPARM, WEB READNEXT QUERYPARM, and WEB ENDBROWSE QUERYPARM) to browse through all the query parameters in a URL.

CICS only reads query string data when CICS is the HTTP server. The facility is not available when CICS is an HTTP client.

Options

QUERYPARM(*data-value***)**
Specifies the name of the query parameter to extract. Specify a string of text containing the name of the requested parameter. The string of text supplied is not case-sensitive. If you specify the HOSTCODEPAGE option, you must supply the name of the query parameter in the code page that you specify for that option.

HOSTCODEPAGE(*data-value***)**
Specifies the 8-character name of the CICS (host) code page required by the application program. This code page is normally an EBCDIC code page. CICS converts the value of the query parameter into this code page before returning it as the VALUE option.

The standard CICS form of a host code page name consists of the code page number (or more generally CCSID) written using 3 to 5 decimal digits as necessary then padded with trailing spaces to 8 characters. For code page 37, which is fewer than 3 digits, the standard form is 037. CICS also accepts any decimal number of up to 8 digits (padded with trailing spaces) in the range 1 to 65535 as a code page name, even if it is not in the standard form.

If the code page is not specified, the data is returned in the EBCDIC code page specified by the LOCALCCSID system initialization parameter (which applies to the local CICS region, and has a default of 037), provided that the specified code page is supported by the CICS web interface. The code page is supported if it is one of a list of EBCDIC code pages that are recognized by CICS as being sufficiently standard to allow successful parsing of the web headers (this includes all SBCS CECP and Euro code pages). Otherwise, CICS returns the data in the default EBCDIC code page 037 instead.

NAMELENGTH(*data-value***)**
Specifies the length, as a fullword binary value, of the query parameter name.

SET(*ptr-ref***)**
Specifies a pointer reference that is to be set to the address of data received.

VALUE(*data-area*)
Specifies the buffer to contain the value of the named query parameter. CICS unescapes any escaped characters before placing them in the buffer.

VALUELENGTH(*data-area*)
Specifies the length, as a fullword binary value, of the query parameter value. The real length of the value is returned in this data area. If you specify the VALUE option, VALUELENGTH specifies the maximum length of the data that the program accepts. If the value exceeds the length of the buffer, the value is truncated. If the length of the query parameter value is less than the size of the buffer, the query parameter value is placed in the leftmost byte positions.

Conditions

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request.

12 The host code page cannot be found.

13 No keyword parameters were supplied in the HTTP request.

14 The code page combination for client and server is invalid.

17 Invalid keyword parameters found in the HTTP request.

22 LENGERR
RESP2 values are:

1 The length in NAMELENGTH or VALUELENGTH is less than or equal to zero.

5 The keyword parameter value has been truncated during a read operation because the receiving buffer is too small.

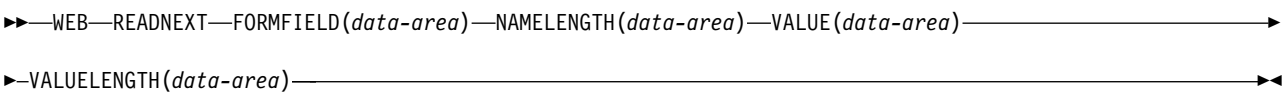
13 NOTFND
RESP2 value is:

1 The keyword parameter with the given name could not be found.

WEB READNEXT FORMFIELD

Retrieve next name-value pair in an HTML form.

WEB READNEXT FORMFIELD



Conditions: ENDFILE, INVREQ, LENGERR

This command is threadsafe.

Description

WEB READNEXT FORMFIELD retrieves the next name-value pair in an HTML form.

The data is returned in its unescaped form (see Reserved and excluded characters for an explanation of this).

Options

FORMFIELD(data-area)
specifies the buffer to contain the name of the form field being retrieved. The case of the name is as it is stored in the form.

NAMELENGTH(data-area)
specifies the length, as a fullword binary value, of the form field name. The actual length of the name is returned in this data area. If the length of the form field name is less than the size of the buffer, the form field name is placed in the leftmost byte positions.

VALUE(data-area)
specifies the buffer to contain the value corresponding to the name returned in the FORMFIELD data area. CICS unescapes any escaped characters before placing them in the buffer.

VALUELENGTH(data-area)
specifies the length, as a fullword binary value, of the form field value. The actual length of the value is returned in this data area. If the value exceeds the buffer length, it is truncated. If the length of the form field value is less than the size of the buffer, the form field value is placed in the leftmost byte positions.

Conditions

20 ENDFILE
The end of the list of name-value pairs has been reached.

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request.

4 The command is being issued before a WEB STARTBROWSE FORMFIELD has been issued.

6 A form field has been found that is not in the format NAME:VALUE.

153 The form type is unknown.

154 A boundary string was expected in the forms data, but was not found.

22 LENGERR
RESP2 values are:

1 NAMELENGTH or VALUELENGTH is less than or equal to zero.

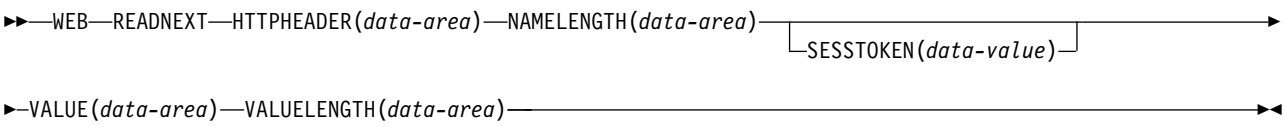
4 The form field name has been truncated during a browse operation because the receiving buffer is too small.

5 The form field value has been truncated because the receiving buffer is too small.

WEB READNEXT HTTPHEADER

Retrieve next HTTP header.

WEB READNEXT HTTPHEADER



Conditions: ENDFILE, INVREQ, LENGERR, NOTOPEN

This command is threadsafe.

Description

WEB READNEXT HTTPHEADER retrieves the next HTTP header in the list of headers. The SESSTOKEN option is required if the HTTP header information is part of a response sent to CICS as an HTTP client.

Options

HTTPHEADER*(data-area)*
specifies the buffer to contain the name of the HTTP header being extracted.

NAMELENGTH*(data-area)*
specifies the length of the buffer supplied on the HTTPHEADER option, as a fullword binary data area, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

SESSTOKEN*(data-value)*
For CICS as an HTTP client, this option is required. It specifies the session token, an 8-byte binary value that uniquely identifies a connection between CICS and a server. This value is returned by a WEB OPEN command for CICS as an HTTP client. Session tokens explains the use of the session token.

VALUE*(data-area)*
specifies the buffer to contain the value of the HTTP header being extracted.

VALUELENGTH*(data-area)*
specifies the length of the buffer supplied on the VALUE option, as a fullword binary data area, and is set to the actual length of the data returned to the application. If the data exceeds the buffer length, a LENGERR condition is raised and the data is truncated.

Conditions

20 ENDFILE
The end of the list of HTTP headers has been reached.

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request.

4 The command is being issued before a WEB STARTBROWSE has been issued.

6 A header has been found which is not in the format NAME:VALUE.

22 LENGERR
RESP2 values are:

1 The length in NAMELENGTH or VALUELENGTH is less than or equal to zero (CICS as an HTTP server).

4 The header name has been truncated because the receiving buffer is too small (CICS as an HTTP server).

5 The header value has been truncated because the receiving buffer is too small (CICS as an HTTP server).

35 The length in NAMELENGTH is less than or equal to zero (CICS as an HTTP client).

51 The header name has been truncated because the receiving buffer is too small (CICS as an HTTP client).

52 The header value has been truncated because the receiving buffer is too small (CICS as an HTTP client).

55 The length in VALUELENGTH is less than or equal to zero (CICS as an HTTP client).

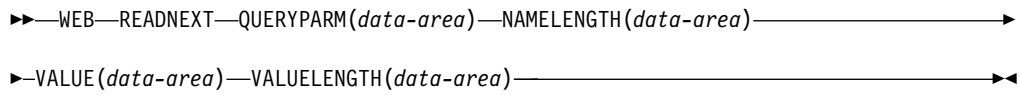
19 NOTOPEN
RESP2 value is:

27 Invalid session token.

WEB READNEXT QUERYPARM

Retrieve next name and value pair in query string data in a URL.

WEB READNEXT QUERYPARM



Conditions: ENDFILE, INVREQ, LENGERR

This command is threadsafe.

Description

WEB READNEXT QUERYPARM retrieves the next keyword parameter (name and value pair) in a query string in a URL.

The data is returned in its unescaped form (see Reserved and excluded characters for an explanation of this).

Options

QUERYPARM(*data-area*)
Specifies the buffer to contain the name of the keyword parameter being retrieved. The case of the name is as it is stored in the keyword parameter.

NAMELENGTH(*data-area*)
Specifies the length, as a fullword binary value, of the keyword parameter name. The actual length of the name is returned in this data area. If the length

of the keyword parameter name is less than the size of the buffer, the keyword parameter name is placed in the leftmost byte positions.

VALUE*(data-area)*
Specifies the buffer to contain the value corresponding to the name returned in the QUERYPARM data area. CICS unescapes any escaped characters before placing them in the buffer.

VALUELENGTH(data-area)
Specifies the length, as a fullword binary value, of the keyword parameter value. The actual length of the value is returned in this data area. If the value exceeds the buffer length, it is truncated. If the length of the keyword parameter value is less than the size of the buffer, the keyword parameter value is placed in the leftmost byte positions.

Conditions

20 ENDFILE
The end of the list of keyword parameters has been reached.

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request.

4 The command is being issued before a WEB STARTBROWSE has been issued.

6 A keyword parameter has been found which is not in the format NAME=VALUE.

22 LENGERR
RESP2 values are:

1 NAMELENGTH or VALUELENGTH is less than or equal to zero.

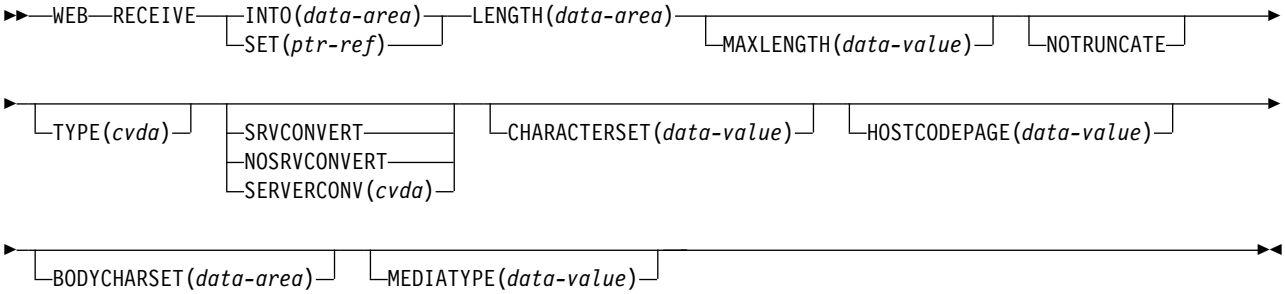
4 The keyword parameter name has been truncated during a browse operation because the receiving buffer is too small.

5 The keyword parameter value has been truncated because the receiving buffer is too small.

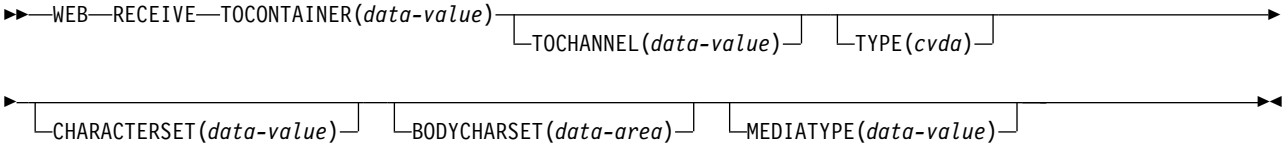
WEB RECEIVE (Server)

Receive an HTTP request, or a non-HTTP message.

WEB RECEIVE (CICS as an HTTP server using buffers)



WEB RECEIVE (CICS as an HTTP server using containers)



Conditions: CHANNELERR, CONTAINERERR, INVREQ, LENGERR, NOTFND

This command is threadsafe.

Description

WEB RECEIVE receives the body of an HTTP request, or all the data for a non-HTTP message, into an application-supplied buffer, or set buffer. Alternatively, an HTTP request can be stored in a named container. The headers for an HTTP request can be examined separately using the WEB HTTPHEADER commands. The item received by the WEB RECEIVE command can be:

- The body of an HTTP request that a Web client has made to CICS as an HTTP server. For guidance on the correct use of the WEB RECEIVE command for this purpose, see Receiving the entity body of an HTTP request .
- A non-HTTP message handled by CICS Web support facilities, with the user-defined (USER) protocol on the TCPIPSERVICE definition. For guidance on non-HTTP messages, see CICS web support and non-HTTP requests .
- A request from another application that has used the CICS business logic interface to contact the application program directly, rather than going through the CICS HTTP listener. For guidance on the CICS business logic interface, see Introduction to the CICS business logic interface.

The data is returned in its escaped form.

When receiving the HTTP body into an application buffer (using either the INTO or SET options), **WEB RECEIVE** allows you to specify the type of code page conversion used for incoming data received by the CICS application program. If you omit all of the code page conversion options (SERVERCONV, CLNTCODEPAGE, CHARACTERSET, HOSTCODEPAGE), no code page conversion takes place.

Code page conversion is not permitted when receiving an HTTP body into a named container. If the named container was created by the user application before the **WEB RECEIVE** command was issued, the container is deleted and re-created. The

HTTP request Content-Type header media type information determines whether the named container is re-created as a BIT or CHAR container (unless the CHARACTERSET option is specified, in which case CICS assumes that the data stored in the container is encoded in the CHARACTERSET code page). If the media type is:

- a text media type, a CHAR container is created.
- a non-text media type, a BIT container is created.

If the HTTP request does not contain media type information, the default of text media type is assumed.

If a CHAR container is created, the IANA-registered name of the Coded Character Set Identifier (CCSID) for the data's current code page is retrieved from the HTTP request Content-Type header charset. If this information is not provided or not supported by CICS, the default of ISO-8859-1 is assumed.

The charset can be overridden by using the CHARACTERSET option. If CHARACTERSET is specified, a CHAR container is created.

Options LENGTH, MAXLENGTH, NOTRUNCATE, SERVERCONV and HOSTCODEPAGE are not permitted when receiving an HTTP body into a named container.

Containers cannot be used to receive messages that are sent over a user protocol socket.

Options

BODYCHARSET*(data-area)*
Specifies the character set of the HTTP request body.

The name of the character set can consist of up to 40 alphanumeric characters, including appropriate punctuation.
If the HTTP body is received into an application buffer, the character set returned is as follows:

- If the INTO or SET option is specified, and the HTTP body is converted, CICS returns the character set of the HTTP body before conversion.
- If the INTO or SET option is specified, and the HTTP body is not converted, CICS returns the character set specified in the Content-Type header. If character set information is not available, blanks are returned.

If the HTTP body is received into a named container, the character set returned is as follows:

- If the container is a CHAR container, CICS returns the character set of the encoded data.
- If the container is a BIT container, CICS returns blanks.

If the value returned is more than 40 bytes, the data is truncated. If the value returned is less than 40 bytes, the data is padded to the right with blanks.

CHARACTERSET*(data-value)*
Specifies the character set that was used by the Web client for the entity body of the received item. The name of the character set can consist of up to 40 alphanumeric characters, including appropriate punctuation. CICS does not support all the character sets named by IANA. HTML coded character sets lists the IANA character sets that are supported by CICS for code page conversion.

If the HTTP request body is stored in a buffer and the CHARACTERSET option is specified, the SRVCONVERT option is assumed, so code page conversion of the entity body takes place. CICS can identify the character set for the message body, when receiving data into a buffer (using either the INTO or SET options), if you specify SRVCONVERT or HOSTCODEPAGE, or both options (without specifying CHARACTERSET). The description for the SERVERCONV option tells you what happens in this case.

If the HTTP request body is stored in a container and the CHARACTERSET option is specified, CICS assumes that the data stored in the container is encoded in that code page. CHARACTERSET overrides the Content-Type charset of the received data, and sets the CCSID of the container to the specified CHARACTERSET. This means that when the GET CONTAINER command is issued with that container, the data is converted from the CCSID that was set by the CHARACTERSET parameter on the WEB RECEIVE command, to any code page requested by the user.

CLNTCODEPAGE*(data-value)*
This option is supported for upgrade purposes only. CHARACTERSET replaces it. The action taken by CICS is the same for both keywords.

HOSTCODEPAGE*(data-value)*
Specifies the 8-character name of the CICS (host) code page used by the application program, into which the entity body of the received item should be converted from the character set in which it was received from the Web client.

The standard CICS form of a host code page name consists of the code page number (or more generally CCSID) written using 3 to 5 decimal digits as necessary then padded with trailing spaces to 8 characters. For code page 37, which is fewer than 3 digits, the standard form is 037. CICS also accepts any decimal number of up to 8 digits (padded with trailing spaces) in the range 1 to 65535 as a code page name, even if it is not in the standard form.

If you are receiving data into a buffer (and either the INTO or SET option is specified), then the HOSTCODEPAGE option is specified and SRVCONVERT is assumed, so code page conversion of the entity body takes place. Specifying either SRVCONVERT, or CHARACTERSET, or both, and omitting HOSTCODEPAGE, lets CICS determine the host code page.

The default if this option is not specified is the default code page for the local CICS region, as specified in the LOCALCCSID system initialization parameter.

If you are using the TOCONTAINER option, do not specify the HOSTCODEPAGE option.

INTO*(data-area)*
Specifies the buffer that is to contain the data being received. If the HTTP request contains a Content-Length header or was sent in chunks, CICS internally holds the HTTP body in 64-bit (above-the-bar) storage; otherwise CICS internally holds the body in 31-bit (above-the-line) storage. The body is copied from this internal storage to the application buffer that is specified in the **INTO** option.

If the **INTO** option is specified, **MAXLENGTH** must also be specified, with a value greater than zero (otherwise an INVREQ error with RESP2 16 is returned to the application).

LENGTH*(data-area)*
Specifies a fullword binary variable that is set to the amount of data that CICS has returned to the application. Note that this might be slightly less than the

limit that you set using the MAXLENGTH option, especially if a double-byte or multi-byte character set is involved, because CICS does not return a partial character at the end of the data.

- If the NOTRUNCATE option **is not** specified, any further data present in the message has now been discarded. A LENGERR response with a RESP2 value of 57 is returned if further data was present.
- If the NOTRUNCATE option **is** specified, any additional data is retained. A LENGERR response with a RESP2 value of 36 is returned if additional data is available. The description for the NOTRUNCATE option tells you what to do in this case.

If you are using an application buffer to store your HTTP body, the LENGTH option must be specified when the INTO or SET options are used. If you are using a named container to store your HTTP body (and therefore specify the TOCONTAINER option), do not use the LENGTH option.

MAXLENGTH*(data-value)*
Specifies the maximum amount, as a fullword binary value, of data that CICS is to pass to the application. The MAXLENGTH option applies whether the INTO or the SET option is specified for receiving data. If the data has been sent using chunked transfer-coding, CICS assembles the chunks into a single message before passing it to the application, so the MAXLENGTH option applies to the total length of the chunked message, rather than to each individual chunk. The data is measured after any code page conversion has taken place. If the length of data exceeds the value specified and the NOTRUNCATE option **is not** specified, the data is truncated to that value, and the remainder of the data is discarded. If the length of data exceeds the value specified and the NOTRUNCATE option **is** specified, CICS retains the remaining data and can use it to satisfy subsequent RECEIVE commands.

If you are using the TOCONTAINER option, do not specify the MAXLENGTH option.

MEDIATYPE ***(data-area)***
Specifies the data content of any message body provided, for example `text/xml`. You must specify a 56-byte area for `MEDIATYPE`. The media type is up to 56 alphanumeric characters, including appropriate punctuation.

NOTRUNCATE
Specifies that when the data available exceeds the length requested on the MAXLENGTH option, the remaining data is not to be discarded immediately but is to be retained for retrieval by subsequent RECEIVE commands. (If no further RECEIVE commands are issued, the data is discarded during transaction termination.)

A single RECEIVE command using the SET option and without the MAXLENGTH option receives all the remaining data, whatever its length. Alternatively, you can use a series of RECEIVE commands with the NOTRUNCATE option to receive the remaining data in appropriate chunks. Keep issuing the RECEIVE command until you are no longer getting a LENGERR response.If you receive less than the length requested on the MAXLENGTH option, this does not necessarily indicate the end of the data; this could happen if CICS needs to avoid returning a partial character at the end of the data.

If you are using the TOCONTAINER option, do not specify the NOTRUNCATE option. The entire HTTP body is stored in the named container by the first **WEB RECEIVE** command.

SERVERCONV(*cvda*)

Specifies whether or not CICS translates the entity body of the item received, from the character set used by the Web client, to a code page suitable for the application. You can use the CHARACTERSET and HOSTCODEPAGE options on this command to specify the character set and code page that are used. If you specify either of these options, code page conversion (SRVCONVERT) is assumed. Alternatively, you can omit either or both of these options, specify SRVCONVERT and let CICS determine a suitable character set and code page.

If you are using the TOCONTAINER option, do not specify the SERVERCONV option.

SRVCONVERT

CICS converts the entity body of the message, before it is passed to the application.

When you specify SRVCONVERT without CHARACTERSET, CICS identifies the character set as follows:

1. If the Web client's request has a Content-Type header naming a character set supported by CICS, that character set is used.
2. If the Web client's request has no Content-Type header or the named character set is unsupported, the ISO-8859-1 character set is used.
3. For non-HTTP messages (sent using the USER protocol), the ISO-8859-1 character set is used.

When you specify SRVCONVERT without HOSTCODEPAGE, CICS determines the host code page as the default code page for the local CICS region, as specified in the LOCALCCSID system initialization parameter.

If you specify SRVCONVERT alone, note that for code page conversion to take place, the media type for the message must specify a type of data content that can be identified as text according to the IANA definitions. For messages where no media type is given but SRVCONVERT is specified, code page conversion also takes place. If a non-text media type is present, CICS does not convert the message body. However, for compatibility with Web-aware applications coded in earlier releases, if you specify either of the CHARACTERSET or HOSTCODEPAGE options or omit the SERVERCONV option, the media type for the message does not influence code page conversion.

NOSRVCONVERT

CICS does not convert the entity body of the item, and it is passed to the application in the character set used by the Web client. If you specify NOSRVCONVERT, you cannot specify the CHARACTERSET or HOSTCODEPAGE options.

SET(*ptr-ref*)

Specifies a pointer reference that is to be set to the address of data received. The pointer reference is valid until the next receive command or the end of task.

If the HTTP request contains a Content-Length header or has been sent in chunks, CICS internally holds the HTTP body in above-the-bar storage; otherwise CICS holds the body in above-the-line storage. The body is copied from this internal storage into storage that is accessible to the application, and the pointer reference is set to this copy.

TOCHANNEL*(data-value)*
Specifies the name of the channel that the container belongs to. The name of the channel can consist of up to 16 alphanumeric characters, including appropriate punctuation. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blanks are not permitted. If the name is less than 16 characters, it is padded with trailing blanks. If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

If you plan to ship your channels between CICS regions, bear in mind that you should restrict your characters to standard alphanumeric characters (A-Z 0-9 & : = , ; <>. - _) to ensure they are represented in the same way by all EBCDIC code pages.

You can specify the channel name DFHTRANSACTION to use a transaction channel. A transaction channel does not go out of scope when the link level changes: it is always accessible in the transaction. For more information, see Channels and containers.

If the TOCHANNEL option is not specified, then CICS assumes the current channel.

TOCONTAINER*(data-value)*
Specifies the name of the container where the data is placed. The name of the container can consist of up to 16 alphanumeric characters, including appropriate punctuation. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blanks are not permitted. If the name is less than 16 characters, it is padded with trailing blanks.

If you plan to ship your containers between CICS regions, bear in mind that you should restrict your characters to standard alphanumeric characters (A-Z 0-9 & : = , ; <>. - _) to ensure they are represented in the same way by all EBCDIC code pages.

Do not use container names starting with "DFH", unless requested to do so by CICS.

The TOCONTAINER option can only be specified on the first **WEB RECEIVE** command.

When TOCONTAINER is specified, the received body of the HTTP request is kept in 64-bit (above-the-bar) storage by CICS, until the application uses a CONTAINER API command to obtain the HTTP body.

TYPE*(cvda)*
Returns the type of request received. CVDA values are:

HTTPYES
indicates an HTTP request.

HTTPNO
indicates a non-HTTP request.

In CICS Transaction Server for z/OS, Version 3, HTTP requests and non-HTTP requests use different protocols, which are specified on TCPIPSERVICE definitions, and must therefore use different ports. Non-HTTP requests use the user-defined (USER) protocol. You might use the TYPE option to distinguish between the request types if you specify the same user-written application program for responding to both HTTP and non-HTTP requests.

Conditions

122 CHANNELERR
RESP2 values are:

1 The name specified by the TOCHANNEL option contains an illegal character or combination of characters.

110 CONTAINERERR
RESP2 values are:

1 The name specified by the TOCONTAINER option contains an illegal character or combination of characters.

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

14 Invalid code page combination.

46 The SERVERCONV option is invalid.

80 CHARACTERSET cannot be specified with NOSRVCONVERT.

81 HOSTCODEPAGE cannot be specified with NOSRVCONVERT.

84 Body incomplete.

145 Channel was not specified, and there is no current channel.

146 The named container is a read-only container.

147 Internal conversion error.

148 User protocol is not supported for containers.

149 The TOCONTAINER option can only be specified on the first WEB RECEIVE command.

22 LENGERR
RESP2 values are:

16 The MAXLENGTH option value was less than or equal to zero.

36 Partial response body returned. Use additional RECEIVEs to obtain remainder.

57 The response body exceeds the length specified, and the remainder of the body has been discarded.

13 NOTFND
RESP2 values are:

7 Code page not found.

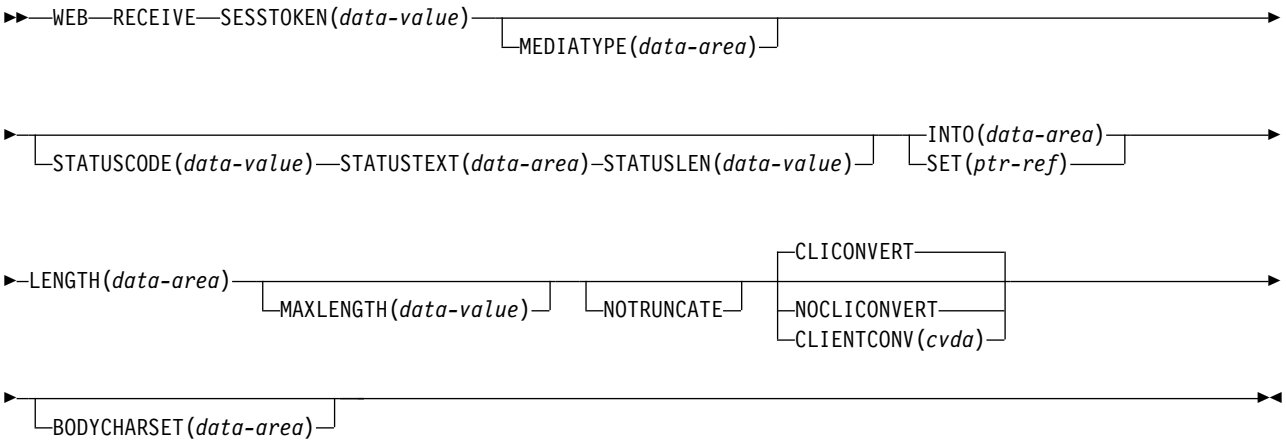
82 Client code page (character set) not found.

83 Host code page (for server) not found.

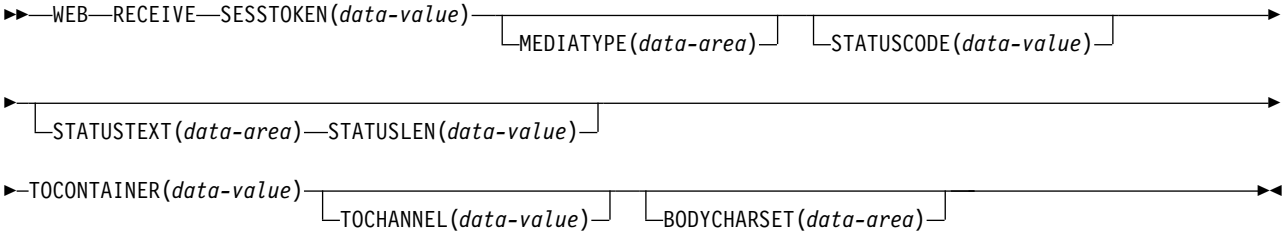
WEB RECEIVE (Client)

Receive an HTTP response for CICS as an HTTP client.

WEB RECEIVE (CICS as an HTTP client using buffers)



WEB RECEIVE (CICS as an HTTP client using containers)



Conditions: CHANNELERR, CONTAINERERR, INVREQ, LENGERR, NOTOPEN, IOERR, TIMEDOUT

This command is threadsafe.

Description

WEB RECEIVE for CICS as an HTTP client receives the body of an HTTP response that a server has made. The headers for the HTTP response can be examined separately using the WEB READ HTTPHEADER command or the HTTP header browsing commands. A session token must be included on this command. For guidance on the correct use of the WEB RECEIVE command for CICS as an HTTP client, see .

When receiving the HTTP body into an application buffer (using either the INTO or SET options), **WEB RECEIVE** allows you to specify the type of code page conversion used for incoming data received by the CICS application program. If you omit all of the code page conversion options (SERVERCONV, CLNTCODEPAGE, CHARACTERSET, HOSTCODEPAGE), no code page conversion takes place.

Code page conversion is not permitted when receiving an HTTP body into a named container. If the named container was created by the user application before the **WEB RECEIVE** command was issued, the container is deleted and re-created. The HTTP request Content-Type header media type information determines whether the named container is re-created as a BIT or CHAR container (unless the

CHARACTERSET option is specified, in which case CICS assumes that the data stored in the container is encoded in the CHARACTERSET code page). If the media type is:

- a text media type, a CHAR container is created.
- a non-text media type, a BIT container is created.

If the HTTP request does not contain media type information, the default of text media type is assumed.

If a CHAR container is created, the IANA-registered name of the Coded Character Set Identifier (CCSID) for the data's current code page is retrieved from the HTTP request Content-Type header charset. If this information is not provided or not supported by CICS, the default of ISO-8859-1 is assumed.

Options LENGTH, MAXLENGTH, NOTRUNCATE and CLIENTCONV are not permitted when receiving an HTTP body into a named container.

Note: The RTIMOUT value specified for the transaction that starts the user application indicates the time that the application is prepared to wait to receive the incoming message. (RTIMOUT is specified on the transaction profile definition). When the period specified by RTIMOUT expires, CICS returns a TIMEDOUT response to the application. An RTIMOUT value of zero means that the application is prepared to wait indefinitely. The default setting for RTIMOUT on transaction profile definitions is zero, so it is important to check and change that setting for applications that are making HTTP client requests.

Options

BODYCHARSET*(data-area)*
specifies the character set of the HTTP response body.

The name of the character set can consist of up to 40 alphanumeric characters, including appropriate punctuation.
If the HTTP body is received into an application buffer, the character set returned is as follows:

- If the INTO or SET option is specified, and the HTTP body is converted, CICS returns the character set of the HTTP body before conversion.
- If the INTO or SET option is specified, and the HTTP body is not converted, CICS returns the character set specified in the Content-Type header. If character set information is not available, blanks are returned.

If the HTTP body is received into a named container, the character set returned is as follows:

- If the container is a CHAR container, CICS returns the character set of the encoded data.
- If the container is a BIT container, CICS returns blanks.

If the value returned is more than 40 bytes, the data is truncated. If the value returned is less than 40 bytes, the data is padded to the right with blanks.

CLIENTCONV*(cvda)*
If you are receiving data into a buffer (and either the INTO or SET option is specified), this option specifies whether or not CICS translates the entity body of the response from the character set used by the server, to a code page suitable for the application. The default is that the entity body is converted.

CLICONVERT
CICS converts the entity body of the response from the character set used by the server, into the code page that you identify for the application.

NOCLICONVERT
CICS does not convert the entity body of the response, and it is passed to the application in the character set used by the server.

You do not need to specify a character set or application code page on the WEB RECEIVE command for CICS as an HTTP client. If code page conversion is required, CICS identifies the character set used by the server by examining the Content-Type header of the message. If the header does not provide this information, or if the named character set is not supported by CICS for code page conversion, the ISO-8859-1 character set is used. For the application's code page, the default code page for the local CICS region (as specified in the LOCALCCSID system initialization parameter) is used, or an alternative EBCDIC code page that you specified on the WEB OPEN command.

There are certain considerations for code page conversion to take place when using buffers (with either the INTO or SET option specified). If you are receiving data into a buffer, and CHARACTERSET and CLICONVERT are not specified, the media type for the message must specify text as the data content type (according to the IANA definitions) for code page conversion to take place. For messages where no media type is given, but CLICONVERT is specified, code page conversion also takes place. If a non-text media type is present, CICS does not convert the message body.

If you are using the TOCONTAINER option, do not specify the CLIENTCONV option.

INTO*(data-area)*
specifies the buffer that is to contain the data being received. If the INTO parameter is specified, then MAXLENGTH must also be specified, with a value greater than zero (otherwise an INVREQ error with RESP2 16 is returned to the application).

LENGTH*(data-area)*
specifies a fullword binary variable which is set to the amount of data that CICS has returned to the application. Note that this might be slightly less than the limit that you set using the MAXLENGTH option, especially if a double-byte or multi-byte character set is involved, because CICS does not return a partial character at the end of the data.

- If the NOTRUNCATE option **is not** specified, any further data present in the message has now been discarded. A LENGERR response with a RESP2 value of 57 is returned if further data was present.
- If the NOTRUNCATE option **is** specified, any additional data is retained. A LENGERR response with a RESP2 value of 36 is returned if additional data is available. The description for the NOTRUNCATE option tells you what to do in this case.

If you are using an application buffer to store your HTTP body, the LENGTH option must be specified when the INTO or SET options are used. If you are using a named container to store your HTTP body (and therefore specify the TOCONTAINER option), do not use the LENGTH option.

MAXLENGTH*(data-value)*
specifies the maximum amount, as a fullword binary value, of data that CICS is to pass to the application. The MAXLENGTH option applies whether the

INTO or the SET option is specified for receiving data. If the data has been sent using chunked transfer-coding, CICS assembles the chunks into a single message before passing it to the application, so the MAXLENGTH option applies to the total length of the chunked message, rather than to each individual chunk. The data is measured after any code page conversion has taken place.

If the length of data exceeds the value specified and the NOTRUNCATE option **is not** specified, the data is truncated to that value, and the remainder of the data is discarded.

If the length of data exceeds the value specified and the NOTRUNCATE option **is** specified, CICS retains the remaining data and can use it to satisfy subsequent RECEIVE commands.

If you are using the TOCONTAINER option, do not specify the MAXLENGTH option.

MEDIATYPE*(data-area)*
specifies the data content of any message body provided, for example `text/xml`. You must specify a 56-byte area for `MEDIATYPE`. The media type is up to 56 alphanumeric characters, including appropriate punctuation. For more information on media types, see IANA media types and character sets.

NOTRUNCATE
specifies that when the data available exceeds the length requested on the MAXLENGTH option, the remaining data is not to be discarded immediately but is to be retained for retrieval by subsequent RECEIVE commands. (If no further RECEIVE commands are issued, the data is discarded during transaction termination.)

A single RECEIVE command using the SET option and without the MAXLENGTH option receives all the remaining data, whatever its length. Alternatively, you can use a series of RECEIVE commands with the NOTRUNCATE option to receive the remaining data in appropriate chunks. Keep issuing the RECEIVE command until you are no longer getting a LENGERR response. If you receive less than the length requested on the MAXLENGTH option, this does not necessarily indicate the end of the data; this could happen if CICS needs to avoid returning a partial character at the end of the data.

If you are using the TOCONTAINER option, do not specify the NOTRUNCATE option. The entire HTTP body is stored in the named container by the first **WEB RECEIVE** command.

SET*(ptr-ref)*
specifies a pointer reference that is to be set to the address of data received. The pointer reference is valid until the next receive command or the end of task.

SESSTOKEN*(data-value)*
specifies the session token, an 8-byte binary value that uniquely identifies a connection between CICS and a server. This value is returned by a WEB OPEN command for CICS as an HTTP client. Session tokens explains the use of the session token.

STATUSCODE*(data-value)*
specifies a data-area to receive the HTTP status code sent by the server. The code is a binary halfword value. Examples are 200 (normal) or 404 (not found). Receiving the status code is optional, but you should always receive and check the status code in the following circumstances:

- If you intend to make an identical request to the server, now or during a future connection.
- If you intend to make further requests to the server using this connection.
- If your application is carrying out any further processing that depends on the information you receive in the response.

HTTP status code reference for CICS web support has basic guidance on appropriate actions for an application to take in response to the status codes for HTTP/1.1.

STATUSTEXT*(data-area)*
specifies a data-area to receive any text returned by the server to describe the status code. The text is known as a reason phrase. Examples are "OK" (accompanying a 200 status code), or "Bad Request" (accompanying a 400 status code). The STATUSLEN option gives the length allowed for the text.

STATUSLEN*(data-value)*
specifies, as a fullword binary value, the length of the data-area to receive any text returned by the server to describe the status code (the STATUSTEXT option). The text is known as a reason phrase. Most reason phrases recommended for HTTP are short, but a data-area length of 256 characters is suggested here, in case the server replaces the recommended reason phrase with more detailed information.

TOCHANNEL*(data-value)*
Specifies the name of the channel that the container belongs to. The name of the channel can consist of up to 16 alphanumeric characters, including appropriate punctuation. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blanks are not permitted. If the name is less than 16 characters, it is padded with trailing blanks. If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

If you plan to ship your channels between CICS regions, bear in mind that you should restrict your characters to standard alphanumeric characters (A-Z 0-9 & : = , ; <>. - _) to ensure they are represented in the same way by all EBCDIC code pages.

You can specify the channel name `DFHTRANSACTION` to use a transaction channel. A transaction channel does not go out of scope when the link level changes: it is always accessible in the transaction. For more information, see Channels and containers.

If the TOCHANNEL option is not specified, then CICS assumes the current channel.

TOCONTAINER*(data-value)*
Specifies the name of the container where the data is placed. The name of the container can consist of up to 16 alphanumeric characters, including appropriate punctuation. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blanks are not permitted. If the name is less than 16 characters, it is padded with trailing blanks.

If you plan to ship your containers between CICS regions, bear in mind that you should restrict your characters to standard alphanumeric characters (A-Z 0-9 & : = , ; <>. - _) to ensure they are represented in the same way by all EBCDIC code pages.

Do not use container names starting with "DFH", unless requested to do so by CICS.

The TOCONTAINER option can only be specified on the first **WEB RECEIVE** command.

Conditions

122 CHANNELERR
RESP2 values are:

1 The name specified by the TOCHANNEL option contains an illegal character or combination of characters.

110 CONTAINERERR
RESP2 values are:

1 The name specified by the TOCONTAINER option contains an illegal character or combination of characters.

19 NOTOPEN
RESP2 values are:

27 Invalid session token.

16 INVREQ
RESP2 values are:

10 Invalid response header.

15 Code page conversion failure.

16 An INTO and MAXLENGTH error has occurred, as a result of one of the following:

- Both parameters, INTO and MAXLENGTH, are not specified.
- Parameters INTO and MAXLENGTH are both specified, but MAXLENGTH is less than or equal to zero.

22 Invalid chunk received.

41 The connection has been closed. The server may have timed out due to inactivity on this connection.

46 The CLIENTCONV option is invalid.

67 The content of the response does not conform to HTTP format. The error is generated because there is a syntax problem.

68 Message send with chunked transfer-coding is in progress.

71 Chunked transfer-coding error.

144 One or more of the Web command parameters is invalid.

145 Channel was not specified, and there is no current channel.

146 The named container is a read-only container.

147 Internal conversion error.

149 The TOCONTAINER option can only be specified on the first WEB RECEIVE command.

22 LENGERR
RESP2 values are:

16 Invalid MAXLENGTH.

36 Partial response body returned. Use additional RECEIVEs to obtain remainder.

57 The response body exceeds the length specified, and the remainder of the body has been discarded.

58 The status text exceeds the length specified and has been truncated.

59 The STATUSLEN option value was not greater than zero.

17 IOERR
RESP2 values are:

42 Socket error.

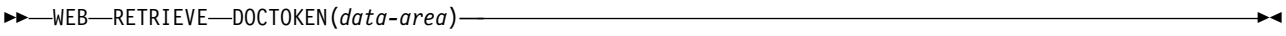
124 TIMEDOUT
RESP2 values are:

62 Timeout on socket receive.

WEB RETRIEVE

Retrieve the DOCTOKEN for a CICS document that was sent using a WEB SEND command.

WEB RETRIEVE



Conditions: INVREQ, NOTFND

This command is threadsafe.

The WEB RETRIEVE command allows an application to retrieve the binary token for a document that was sent. For the WEB RETRIEVE command to be able to retrieve the document token, the previous WEB SEND command must specify the ACTION(EVENTUAL) option, so that the SEND command is pending when the application completes. This is because if the ACTION(EVENTUAL) option is used, the Web domain keeps a copy of the information for a document after it is sent. Note that the document that is retrieved is the document that was sent, and it does not include any changes that might have occurred up to the time when the RETRIEVE command is issued.

A valid sequence of events for issuing the WEB RETRIEVE command is as follows:

```
EXEC CICS WEB SEND
          ACTION(EVENTUAL)
EXEC CICS WEB SEND
          ACTION(EVENTUAL)
EXEC CICS WEB RETRIEVE
          DOCTOKEN(MYDOC)
```

The DOCTOKEN for the second WEB SEND command is retrieved successfully.

If the WEB SEND command specified the option DOCSTATUS(DOCDELETE), the WEB RETRIEVE command cannot retrieve the document, and a NOTFND response with a RESP2 value of 1 is returned.

If the WEB SEND command specified the option ACTION(IMMEDIATE), the WEB RETRIEVE command cannot retrieve the document, and a NOTFND response with a RESP2 value of 1 is returned. WEB SEND client processing does not support ACTION(EVENTUAL), so the WEB RETRIEVE command is not valid for use with WEB SEND (Client) commands.

Options

DOCTOKEN(data-area)
specifies a buffer that contains the 16-byte binary token of the document to be retrieved.

Conditions

16 INVREQ
RESP2 values:

1 The command is issued in a non-CICS Web support application.

2 A WEB SEND command has not been issued..

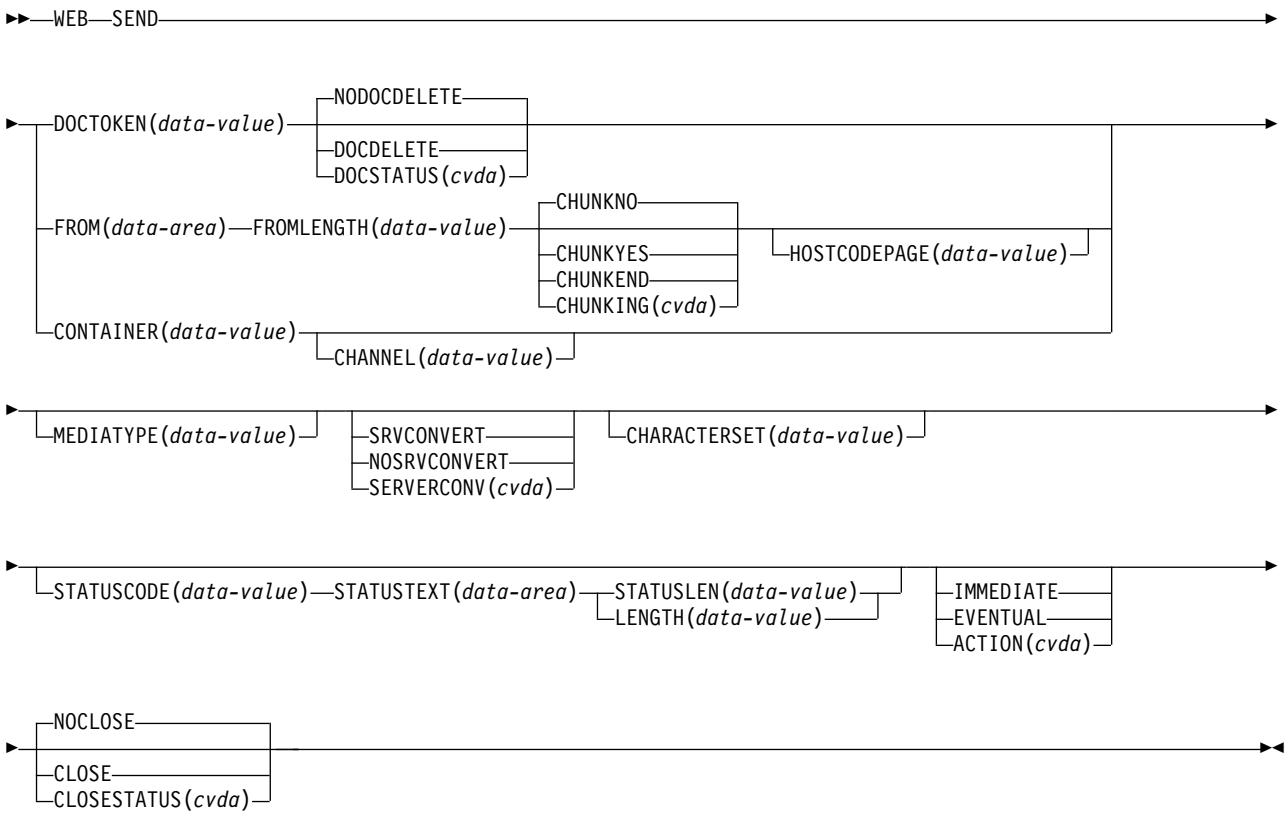
13 NOTFND
RESP2 values:

1 Document is not available as the last WEB SEND specified DOCSTATUS(DOCDELETE) or the last WEB SEND was not a SEND for a document with ACTION(EVENTUAL).

WEB SEND (Server)

Send an HTTP response, or a non-HTTP message.

WEB SEND



Conditions: CHANNELERR, CONTAINERERR, IOERR, INVREQ, LENGERR, NOTFND

This command is threadsafe.

Description

When the CICS application is in the server role, the **WEB SEND** command specifies a response to be sent using CICS Web support or the CICS business logic interface. Here are the possible responses:

- A response to an HTTP request that was made by a Web client, to CICS as an HTTP server. For guidance on the correct use of the WEB SEND command for this purpose, see Sending an HTTP response from CICS as an HTTP server.
- A non-HTTP message handled by CICS Web support facilities, with the user-defined (USER) protocol on the TCPIPSERVICE definition. For guidance on non-HTTP messages, see CICS web support and non-HTTP requests.
- A response to a request from another application that has used the CICS business logic interface to contact the program directly, rather than going through the CICS HTTP listener. For guidance on the CICS business logic interface, see Introduction to the CICS business logic interface.

One response only can be sent during a task. This can be a standard response using one WEB SEND command, or a chunked response using a sequence of WEB SEND commands.

If you attempt to send a second response during the same task, the result depends on whether the IMMEDIATE option or the EVENTUAL option was specified on the WEB SEND command for the first response.

- If the IMMEDIATE option was used for the first response, an error is returned when you attempt the second response.
- If the EVENTUAL option was used for the first response, the second response overwrites the components of the previous response (status line, HTTP headers and message body). The first response is lost, and the second response is sent.

Each time a request from a Web client is received, CICS starts a new task to process the request.

Options

ACTION*(cvda)*

Specifies how the message should be sent out. The CVDA values that apply for CICS as an HTTP server are:

IMMEDIATE
sends the response immediately to the Web client. If CHUNKING is specified, the IMMEDIATE option is assumed.

EVENTUAL
sends the response to the Web client at end of task. If CHUNKING is specified, the EVENTUAL option is ignored. For message sends that do not use chunked transfer-coding, EVENTUAL is the default.

CHANNEL*(data-value)*

Specifies the name of the channel to which the container belongs. The name of the channel can consist of up to 16 alphanumeric characters, including appropriate punctuation. Leading and embedded blanks are not permitted. If the name is less than 16 characters, it is padded with trailing blanks. You can specify the channel name `DFHTRANSACTION` to use the transaction channel.

If the CONTAINER option is specified, CHANNEL is optional.

If the CHANNEL option is not specified, CICS assumes the current channel.

CHARACTERSET*(data-value)*

Specifies a character set into which CICS translates the entity body of the item sent by the command before sending. The name of the character set can consist of up to 40 alphanumeric characters, including appropriate punctuation. CICS does not support all the character sets named by IANA. HTML coded character sets lists the IANA character sets that are supported by CICS for code page conversion.

When the CHARACTERSET option is specified, SRVCONVERT is assumed, so code page conversion of the entity body takes place. As an alternative to selecting the character set yourself, specifying either SRVCONVERT, or HOSTCODEPAGE (if allowed), or both, and omitting CHARACTERSET, lets CICS determine a suitable character set for the message body. The description for the SERVERCONV option tells you what happens in this case.

If you omit all of the code page conversion options, no code page conversion takes place.

CHUNKING*(cvda)*

is used for controlling the message send when the message is being sent in chunks (known as chunked transfer-coding). The default when the option is not specified is that chunked transfer-coding is not in use. Chunked

transfer-coding is only acceptable to HTTP/1.1 clients, and it cannot be used with HTTP/1.0 clients or non-HTTP messages.

The content of a chunked message can be divided into chunks in whatever way is most convenient for the application program. The body of a chunked message cannot be formed directly from CICS documents, so the DOCTOKEN option cannot be used.

Use a separate WEB SEND command with the CHUNKYES option for each chunk of the message. Use the FROM option to specify the chunk of data, and the FROMLENGTH option to specify the length of the chunk. Other options for the message, such as the CLOSESTATUS option, can be specified on the first WEB SEND command of the sequence (which sends the first chunk), but do not specify them on subsequent commands (which send the second and subsequent chunks).

When you have sent the last chunk of the data, specify a further WEB SEND command with the CHUNKEND option and no FROM or FROMLENGTH option. CICS then sends an empty chunk to the recipient to complete the chunked message.

If one of the WEB SEND commands fails during the sequence, an error response is returned, and subsequent sends will also fail. The application should handle this situation appropriately. If all of the chunks are sent successfully but the application does not issue the final WEB SEND command with the CHUNKEND option, the transaction is abended with abend code AWBP. An incomplete chunked message should be ignored and discarded by the recipient.

Sending an HTTP request or response with chunked transfer-coding has a full description of the procedure for chunked transfer-coding, which should be followed in order for your chunked message to be acceptable to the recipient. CVDA values are:

CHUNKNO
Chunked transfer-coding is not used for the message. This is the default if the CHUNKING option is not specified.

CHUNKYES
Chunked transfer-coding is in progress. The data specified by the FROM option represents a chunk of the message.

CHUNKEND
Chunked transfer-coding is complete. No data is specified for this send. CICS sends an empty chunk to the recipient to complete the chunked message.

If you are using the CONTAINER option, do not specify the CHUNKING option. A chunked response cannot be sent from a container.

CLNTCODEPAGE*(data-value)*
This option is supported for upgrade purposes only. CHARACTERSET replaces it. The action taken by CICS is the same for both keywords.

CLOSESTATUS*(cvda)*
Specifies whether or not CICS closes the connection after sending the message. The default is that the connection is not closed. The CVDA values are:

CLOSE
CICS writes a Connection header with the "close" connection option (Connection: close) for this response, and closes the connection with the Web client after sending the response. The header notifies the Web

client of the closure. (For a Web client at HTTP/1.0 level, CICS achieves the same effect by omitting the Connection: Keep-Alive header.)

If chunked transfer-coding is in use, the CLOSE option can be specified on the first chunk of the message, to inform the Web client that the connection is closed after the chunked message is complete.

NOCLOSE
means that the Connection: close header is not used for this response, and the connection is kept open. If the Web client is identified as HTTP/1.0 and has sent a Connection header with the "Keep-Alive" connection option (Connection: Keep-Alive), CICS sends the same header, to notify that a persistent connection will be maintained.

CONTAINER*(data-value)*
Specifies the name of the container where the HTTP body is held, before it is sent to the server. The name of the container can consist of up to 16 alphanumeric characters, including appropriate punctuation. Leading and embedded blanks are not permitted. If the name is shorter than 16 characters, it is padded with trailing blanks.

When CONTAINER is specified, the HTTP body is kept in 64-bit (above-the-bar) storage by CICS and is sent from that storage.

DOCSTATUS*(cvda)*
indicates whether the document should be deleted or not deleted during processing of the WEB SEND command. The CVDA values are:

DOCDELETE
CICS deletes the document after the document contents are saved for sending. Storage allocated for the document is released immediately. If you make subsequent requests for the document, these generate a NOTFND response.

NODOCDELETE
CICS does not delete the document during processing of the WEB SEND command. This is the default value for DOCSTATUS.

DOCTOKEN*(data-value)*
Specifies the 16-byte binary token of a document to be sent as the message body. The document is created using the CICS Document interface using the **EXEC CICS DOCUMENT CREATE**, **INSERT**, and **SET** commands. The FROM option provides an alternative way to create a message body.

The body of a chunked message cannot be formed from CICS documents, so the DOCTOKEN option cannot be used for chunked transfer-coding.

When DOCTOKEN is specified, the document is kept in 31-bit (above-the-line) storage.

FROM*(data-area)*
Specifies a buffer of data which holds the complete message body, or a chunk of the message body. The message body is built by the application program. When you specify the FROM option, use the FROMLENGTH option to specify the length of the buffer of data. The DOCTOKEN option provides an alternative way to create the message body, but that option cannot be used for the body of a chunked message.

When FROM is specified, if CICS needs to either copy or convert the HTTP body before it is sent, the body is copied or converted in 64-bit (above-the-bar) storage, from where it is sent.

There is no set maximum limit for the size of the data-area, but its size is limited in practice by storage considerations. Producing an entity body for an HTTP message has more information about these.

FROMLENGTH*(data-value)*
Specifies the length, as a fullword binary value, of the buffer of data supplied on the FROM option. It is important to state this value correctly, because an incorrect data length can cause problems for the recipient of the message.

HOSTCODEPAGE*(data-value)*
Specifies the 8-character name of the CICS (host) code page that was used by the application program for the entity body of the response.

The standard CICS form of a host code page name consists of the code page number (or more generally CCSID) written using 3 to 5 decimal digits as necessary then padded with trailing spaces to 8 characters. For code page 37, which is fewer than 3 digits, the standard form is 037. CICS now also accepts any decimal number of up to 8 digits (padded with trailing spaces) in the range 1 to 65535 as a code page name, even if it is not in the standard form.

When the HOSTCODEPAGE option is specified, SRVCONVERT is assumed, so code page conversion of the entity body takes place. Specifying either SRVCONVERT, or CHARACTERSET, or both, and omitting HOSTCODEPAGE, lets CICS identify the host code page.

If a CICS document is used to form the response body (DOCTOKEN option), do not specify the HOSTCODEPAGE option, because CICS identifies the host code page from the CICS document domain's record of the host code pages for the document.

If a buffer of data is used to form the response body (FROM option), you may need to specify HOSTCODEPAGE. The default if this option is not present is the default code page for the local CICS region, as set in the LOCALCCSID system initialization parameter. If you require code page conversion but your application has used a different code page, use HOSTCODEPAGE to specify it.

If you omit all of the code page conversion options, no code page conversion takes place.

If you are using the CONTAINER option, do not specify the HOSTCODEPAGE option.

LENGTH*(data-value)*
This option is supported for upgrade purposes only. STATUSLEN replaces it.

MEDIATYPE*(data-value)*
Specifies the data content of any message body provided, for example `text/xml`. You must specify a 56-byte area for `MEDIATYPE`. The media type is up to 56 alphanumeric characters, including appropriate punctuation, but not spaces. For more information on media types, see IANA media types and character sets. CICS checks that the format of the media type is correct, but does not check the validity of the media type against the data content. CICS does not provide a default. In some circumstances, the media type that you specify affects whether or not code page conversion is carried out; see the description of the SERVERCONV option for more information.

SERVERCONV*(cvda)*
Specifies whether or not CICS translates the entity body of the item sent by the command before sending, from the code page used by the application, to a character set suitable for the recipient. You can use the CHARACTERSET and HOSTCODEPAGE options on this command to specify the character set and

code page that are used. If you specify either of these options, code page conversion (SRVCONVERT) is assumed. Alternatively, you can omit either or both of these options, specify SRVCONVERT and let CICS determine a suitable character set and code page.

SRVCONVERT
CICS converts the entity body of the message.

When you specify SRVCONVERT without CHARACTERSET, CICS determines a suitable character set as follows:

1. If the Web client's request has a Content-Type header naming a character set supported by CICS, that character set is used.
2. If the Web client's request has no Content-Type header or the named character set is unsupported, the ISO-8859-1 character set is used.
3. For non-HTTP messages (sent using the USER protocol), the ISO-8859-1 character set is used.

When you specify SRVCONVERT without HOSTCODEPAGE, CICS identifies the host code page as follows:

- If the FROM option is used, CICS identifies the host code page as the default code page for the local CICS region, as specified in the LOCALCCSID system initialization parameter.
- If the DOCTOKEN option is used, CICS identifies the host code page from the CICS document domain's record of the host code pages for the document.
- If the CONTAINER option is used, CICS identifies the host code page as the code page that was used for data encoding, when the HTTP body was stored in the container.

If you specify SRVCONVERT alone, note that for code page conversion to take place, the MEDIATYPE option must specify a type of data content that can be identified as text according to the IANA definitions. For non-text media types, CICS does not convert the message body, and an INVREQ RESP2 code is issued. However, for compatibility with Web-aware applications coded in earlier releases, if you specify either of the CHARACTERSET or HOSTCODEPAGE options or omit the SERVERCONV option, the MEDIATYPE option does not influence code page conversion.

BIT containers contain non-text media, and therefore do not support code page conversion. As a result, if you code either the SRVCONVERT or CHARACTERSET options with a BIT container, an INVREQ RESP2 error is produced.

NOSRVCONVERT
CICS does not convert the entity body of the HTTP request, and it is sent to the server in the code page used by the application. If you specify NOSRVCONVERT, you cannot specify the CHARACTERSET or HOSTCODEPAGE options.

Note: If you omit all of the code page conversion options (SERVERCONV, CLNTCODEPAGE, CHARACTERSET, HOSTCODEPAGE), no code page conversion takes place.

STATUSCODE*(data-value)*
Specifies a standard HTTP status code determined by the application program, which is to be inserted on the status line of the HTTP response. The code is a

halfword binary value. Examples are 200 (normal response) or 404 (not found). If this option is not specified, CICS supplies a default of 200.

HTTP status code reference for CICS web support has information about the use of status codes for CICS Web support. For status codes 204, 205, and 304, a message body is not allowed, and CICS returns an error response to the command if you attempt to include one. Other than that, CICS does not check that your use of the status code is appropriate.

STATUSLEN*(data-value)*
Specifies the length, as a fullword binary value, of the string supplied on the STATUSTEXT option.

STATUSTEXT*(data-area)*
Specifies a data-area containing human-readable text to describe the reason for the status code. The text is known as a reason phrase. Examples are "OK" (accompanying a 200 status code), or "Bad Request" (accompanying a 400® status code). The HTTP/1.1 specification (RFC 2616) defines a recommended reason phrase for each status code, but you do not have to use these.

Conditions

122 CHANNELERR
RESP2 values are:

2 The channel specified by the CHANNEL option could not be found.

110 CONTAINERERR
RESP2 values are:

2 The container specified by the CONTAINER option could not be found.

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

11 Action code invalid.

13 Close status invalid.

14 Invalid code page combination.

32 Media type invalid.

41 The connection has been closed.

46 The SERVERCONV option is invalid.

72 Status code does not support a message body.

75 Invalid send sequence.

77 Chunk incomplete.

80 CHARACTERSET cannot be specified with NOSRVCONVERT.

81 HOSTCODEPAGE cannot be specified with NOSRVCONVERT.

85 Chunking cannot be used with non-HTTP messages.

86 Chunking cannot be used with HTTP/1.0 clients.

87 Status code not allowed.

88 Host code page not allowed.

89 Previous send over this connection failed. No further sends permitted.

90 STATUSCODE and STATUSTEXT options not allowed for second or subsequent chunks.

91 CHARACTERSET and CLNTCODEPAGE options not allowed for second or subsequent chunks.

92 HOSTCODEPAGE option not allowed for second or subsequent chunks.

93 MEDIATYPE option not allowed for second or subsequent chunks.

94 CLOSESTATUS option not allowed for second or subsequent chunks.

95 SERVERCONV option not allowed for second or subsequent chunks.

120 The CHUNKING option is invalid.

121 FROMLENGTH option required.

122 FROM option required.

123 No message body specified. Use FROM, DOCTOKEN or CHUNKEND.

124 CHUNKING option not specified, FROMLENGTH option required.

125 CHUNKNO specified, FROM option required.

126 CHUNKNO specified, FROMLENGTH option required.

127 CHUNKYES specified, FROM option required.

128 CHUNKYES specified, FROMLENGTH option required.

129 FROM option not allowed with CHUNKEND.

130 FROMLENGTH option not allowed with CHUNKEND.

131 FROMLENGTH option specified as zero.

143 The DOCSTATUS value specified is invalid.

145 Channel was not specified, and there is no current channel.

147 Internal conversion error.

148 User protocol is not supported for containers.

150 Conversion requested, but data to be sent is in a DATATYPE BIT container.

151 Chunking is invalid during Web error processing.

152 ACTION(EVENTUAL) is invalid during Web error processing.

17 IOERR

RESP2 values are:

42 Socket error.

22 LENGERR

RESP2 values are:

50 The FROMLENGTH option value was not greater than zero.

13 NOTFND

RESP2 values are:

1 The document has not been created, or has been deleted, or the name is incorrectly specified.

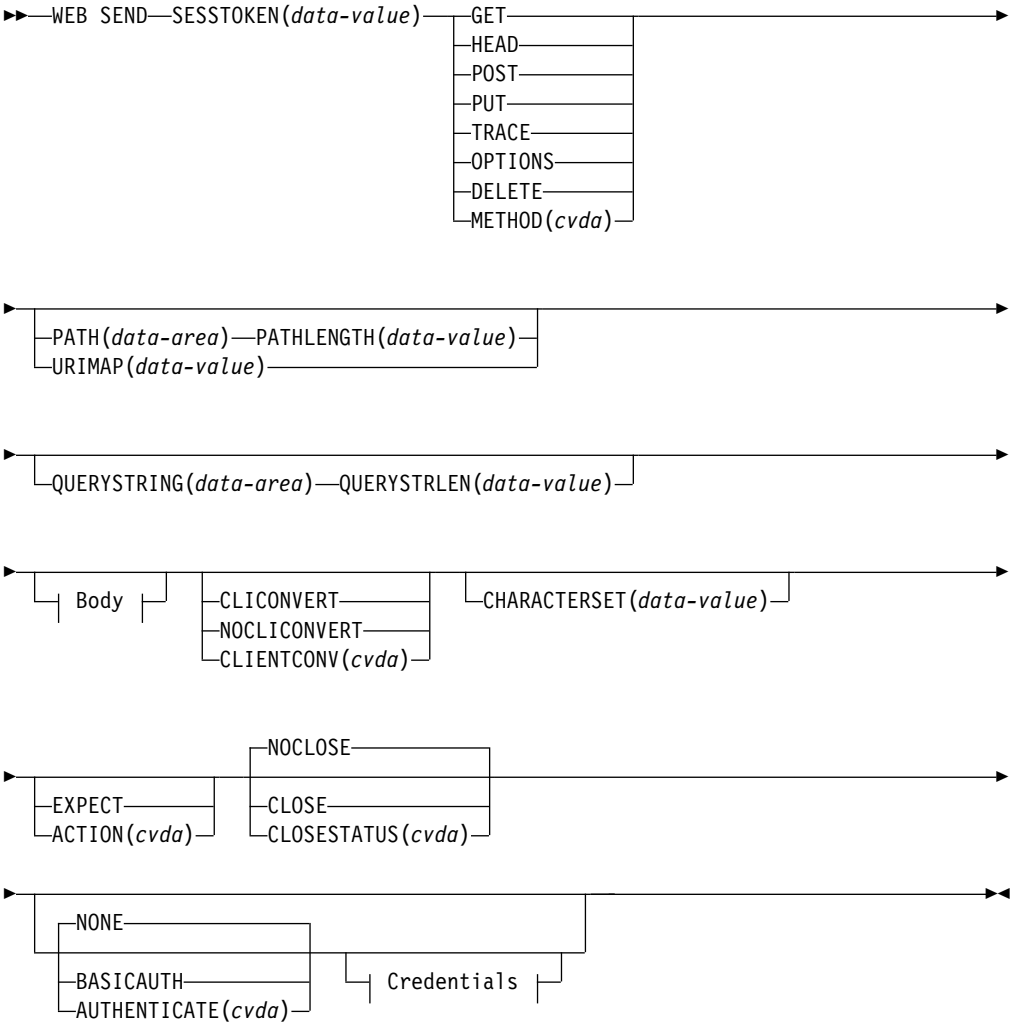
7 Client code page (character set) not found.

83 Host code page (for server) not found.

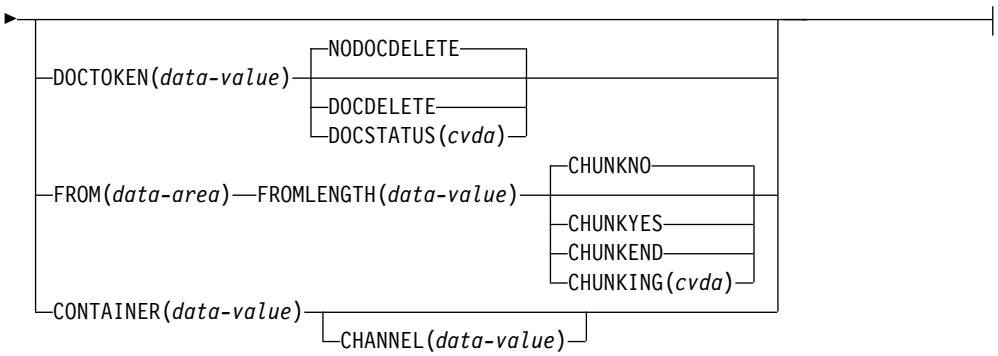
WEB SEND (Client)

Send an HTTP request by CICS as an HTTP client, using CICS Web support.

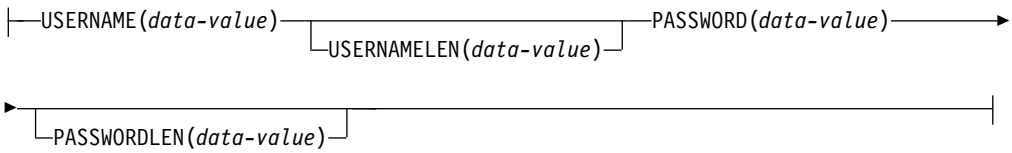
WEB SEND (CICS as an HTTP client)



Body:



Credentials:



Conditions: CHANNELERR, CONTAINERERR, INVREQ, IOERR, LENGERR, NOTAUTH, NOTFND, NOTOPEN, TIMEDOUT, TOKENERR

This command is threadsafe.

Description

WEB SEND for CICS as an HTTP client is used to make an HTTP request to a server. A session token must be included on this command. For guidance on the correct use of the WEB SEND command for CICS as an HTTP client, see .

For CICS as an HTTP client, the WEB SEND command cannot be used after the connection to the server has been closed. You might encounter this situation if either the application program or the server sends a Connection: close header on a message. If you need to test whether the server has requested termination of the connection, use the WEB READ HTTPHEADER command to look for the Connection: close header in the last message from the server.

For CICS as an HTTP client, the CONVERSE command can be used as an alternative to issuing a WEB SEND command followed by a WEB RECEIVE command. However, note that the WEB CONVERSE command does not support chunked transfer-coding, because it requires a sequence of send actions, and the WEB CONVERSE command provides a single send action.

The request can time out when sending a message to the server. In this case, the deadlock time out interval specified in the DTIMOUT attribute of the TRANSACTION definition applies, and CICS returns a TIMEDOUT response to the application.

Options

ACTION*(cvda)*
Specifies how the message will be sent out. This CVDA value applies for CICS as an HTTP client:

EXPECT
Makes CICS send an Expect header with the request line and headers for the request and await a 100-Continue response before sending the message body to the server. If a response other than 100-Continue is received, CICS informs the application program and cancels the send. If no response is received after a period of waiting, CICS sends the message body.

The Expect header is not supported by servers below HTTP/1.1. If CICS does not yet know the HTTP version of the server, CICS makes an additional request before sending your request, to determine the HTTP version of the server. If the Expect header is not suitable, CICS sends your request without it.

This option must be used only if your request has a message body.

AUTHENTICATE*(cvda)*
Specifies user authentication details, to control access to restricted data. The CVDA values that apply for CICS as an HTTP client are as follows:

NONE
Specifies that there are no restrictions on accessing this data, therefore no credentials are required. This is the default value for AUTHENTICATE.

BASICAUTH
Specifies that HTTP Basic Authentication credentials are required for this session. These details can be supplied within the command or by using the XWBAUTH global user exit.

CHANNEL*(data-value)*
Specifies the name of the channel to which the container belongs. The name of the channel can consist of up to 16 alphanumeric characters, including appropriate punctuation. Leading and embedded blanks are not permitted. If the name is less than 16 characters, it is padded with trailing blanks. You can specify the channel name `DFHTRANSACTION` to use the transaction channel.

If the CONTAINER option is specified, CHANNEL is optional.

If the CHANNEL option is not specified, CICS assumes the current channel.

CHARACTERSET*(data-value)*
Specifies the character set into which CICS translates the entity body of the request before sending. The name of the character set can consist of up to 40 alphanumeric characters, including appropriate punctuation. CICS does not support all the character sets named by IANA. HTML coded character sets lists the IANA character sets that are supported by CICS for code page conversion.

For conversion of the entity body to take place, the CLIENTCONV option must be specified as (or allowed to default to) CLICONVERT. Specifying NOCLICONVERT suppresses conversion of the entity body. If conversion is requested, ISO-8859-1 is used as the default if the CHARACTERSET attribute is not specified.

CHUNKING*(cvda)*
Is used for controlling the message send when the message is being sent in chunks (known as chunked transfer-coding). The default when the option is not specified is that chunked transfer-coding is not in use.

The content of a chunked message can be divided into chunks to suit the application program. The body of a chunked message cannot be formed directly from CICS documents, so the DOCTOKEN option cannot be used.

Use a separate WEB SEND command with CHUNKYES for each chunk of the message. Use the FROM option to specify the chunk of data, and the FROMLENGTH option to specify the length of the chunk. Other options for the message, such as the CLOSESTATUS option, can be specified on the first WEB SEND command of the sequence (which sends the first chunk), but do not specify them on subsequent commands (which send the second and subsequent chunks).

When you have sent the last chunk of the data, specify a further WEB SEND command with CHUNKEND, but not the FROM and FROMLENGTH options. CICS then sends an empty chunk to the recipient to end the chunked message.

If your application program is informed of an error at any point in the chunking process, use the WEB CLOSE command to stop the process and close the connection. The recipient of the chunked message will not receive the final empty chunk, and so ignores and discards the data that you have sent so far.

Sending an HTTP request or response with chunked transfer-coding has a full description of the procedure for chunked transfer-coding, which must be followed for your chunked message to be acceptable to the recipient. CVDA values are as follows:

CHUNKNO
Chunked transfer-coding is not used for the message. CHUNKNO is the default if the CHUNKING option is not specified.

CHUNKYES
Chunked transfer-coding is in progress. The data specified by the FROM option represents a chunk of the message.

CHUNKEND
Chunked transfer-coding is complete. No data is specified for this send. CICS sends an empty chunk to the recipient to complete the chunked message.

Notes:

1. The method (METHOD option) must be compatible with chunked transfer-coding.
2. When you have begun sending the parts of a chunked message, the application program cannot send any different messages or receive any items until the final empty chunk is sent and the chunked message is complete.

If you are using the CONTAINER option, do not specify the CHUNKING option. A chunked response cannot be sent from a container.

CLOSESTATUS*(cvda)*
Specifies whether a Connection header with the "close" connection option (Connection: close) will be included on the message. The default is that the header is not included. The CVDA values are as follows:

CLOSE
Makes CICS write a Connection: close header for this request. The header notifies the server that the connection will be closed after the server has sent its response to the request. (For a server at HTTP/1.0 level, CICS achieves the same effect by omitting the Connection: Keep-Alive header.) Do not specify this option if you have implemented connection pooling in the URIMAP resource for this connection, because a closed connection cannot be pooled for reuse.

Only specify this option if this is your final request to the server and you are not using connection pooling.

When you specify the CLOSE option on a WEB SEND command, no further messages can be sent to the server until a new connection is made. The exception is where chunked transfer-coding is in use, when you can specify the CLOSE option on the first chunk of the message, to inform the server that the connection will be closed after the chunked message is complete and a response has been sent.

NOCLOSE
Means that the Connection: close header is not used for this request. If the server is identified as HTTP/1.0, CICS sends a Connection header with the Keep-Alive connection option (Connection: Keep-Alive), to notify that a persistent connection is required.

CLIENTCONV*(cvda)*
Specifies whether CICS translates the entity body of the HTTP request before sending, from the code page used by the application, to a character set suitable for the recipient. If this option is omitted, the default is that any entity body is converted, unless a nontext media type is specified. CVDA values are as follows:

CLICONVERT
CICS converts the entity body of the HTTP request from the code page used by the application, into the character set that you identify for the server. You can use the CHARACTERSET option on this command to specify the character set that is used. If conversion is requested but you do not specify a character set, the default is that CICS converts the entity body to the ISO-8859-1 character set. (The code page used by the application was identified on the WEB OPEN command for the connection.)

For non-text media types, CICS only converts the message body under the following circumstances:

- The message body is sent from a buffer, using the FROM option, and either the CLICONVERT or CHARACTERSET options, or both, are specified.
- The message body is sent from a document, using the DOCTOKEN option.
- The message body is sent from a container and the CHARACTERSET option is specified.

BIT containers contain nontext media, and therefore do not support code page conversion. As a result, if you code either the CLICONVERT or CHARACTERSET options with a BIT container, an INVREQ RESP2 error is produced.

NOCLICONVERT
CICS does not convert the entity body of the HTTP request, and it is sent to the server in the code page used by the application, as identified on the WEB OPEN command for the connection.

CONTAINER*(data-value)*
Specifies the name of the container where the HTTP body is held, before it is sent to the server. The name of the container can consist of up to 16 alphanumeric characters, including appropriate punctuation. Leading and embedded blanks are not permitted. If the name is shorter than 16 characters, it is padded with trailing blanks.

DOCSTATUS*(cvda)*

Indicates whether the document will be deleted or not deleted during processing of the WEB SEND command. The CVDA values are as follows:

DOCDELETE

CICS deletes the document after the document contents are saved for sending. Storage allocated for the document is released immediately. If you make subsequent requests for the document, the requests generate a TOKENERR response.

NODOCDELETE

CICS does not delete the document during processing of the WEB SEND command. This value is the default for DOCSTATUS.

DOCTOKEN*(data-value)*

Specifies the 16-byte binary token of a document to be sent as the message body. You create the document using the CICS Document interface (EXEC CICS DOCUMENT CREATE, INSERT, and SET commands). You do not have to retrieve the document before sending it. The FROM option provides an alternative way to create a message body.

The body of a chunked message cannot be formed from CICS documents, so the DOCTOKEN option cannot be used for chunked transfer-coding.

FROM*(data-area)*

Specifies a buffer of data, which holds the message body. The message body is built by the application program. When you specify the FROM option, use the FROMLENGTH option to specify the length of the buffer of data. The DOCTOKEN and CONTAINER options provide an alternative way to create the message body, but the DOCTOKEN option cannot be used for the body of a chunked message.

The size of the data-area has no set maximum limit, but its size is limited in practice by storage considerations. Producing an entity body for an HTTP message has more information about storage considerations.

FROMLENGTH*(data-value)*

Specifies the length, as a fullword binary value, of the buffer of data supplied on the FROM option (the message body). You must state this value correctly, because an incorrect data length can cause problems for the recipient of the message.

MEDIATYPE*(data-value)*

Specifies the data content of any message body provided, for example `text/xml`. You must specify a 56-byte area for `MEDIATYPE`. The media type is up to 56 alphanumeric characters, including appropriate punctuation, but not spaces. For more information on media types, see IANA media types and character sets. CICS checks that the format of the media type is correct, but does not check the validity of the media type against the data content. CICS uses this information to produce the Content-Type header for the message.

For requests that require a body, you must specify the MEDIATYPE option. There is no default. However, if the required Content-Type header must contain spaces or more than 56 characters, the application can provide it using the WEB WRITE HTTPHEADER command. In this case, do not specify the MEDIATYPE option.

The supplied media type is used to determine whether code page conversion is required under the following circumstances:

- If you are sending a message from a buffer, using the FROM option, and the CLIENTCONV and CHARACTERSET options are not specified.

- If you are sending a message from a document, using the DOCTOKEN option, and the CLIENTCONV and CHARACTERSET options are not specified.
- If you are sending a message from a named container, using the CONTAINER option, and either CLICONVERT is specified, or the CLIENTCONV and CHARACTERSET options are not specified.

If the supplied media type is text, the message is converted. If the supplied media type is nontext, the message is not converted.

METHOD*(cvda)*
Specifies the HTTP method for the request.

The GET, HEAD, POST, PUT, TRACE, OPTIONS, and DELETE methods are supported by this command. However, some HTTP servers, particularly HTTP/1.0 servers, might not implement all of these methods.

HTTP method reference for CICS web support has more information about the correct use of methods, including the HTTP versions that apply to each.

CICS prevents the sending of a message body for methods for which it is inappropriate, and requires it for methods where it is appropriate. Chunked transfer-coding is not relevant for methods that do not have a request body.CVDA values are as follows:

GET Obtain a resource from the server. A request body is not allowed.

HEAD
Obtain the HTTP headers, but not the response body, for a resource. A request body is not allowed.

POST Send data to a server. A request body is required.

PUT Create or modify a resource on the server. A request body is required.

TRACE
Trace the route of your request to the server. A request body is not allowed.

OPTIONS
Obtain information about the server. A request body is allowed, but the body has no defined purpose. If you do use a request body, then you must specify a media type.

DELETE
Delete a resource on the server. A request body is not allowed.

PASSWORD*(data-value)*
Specifies the password associated with the user ID or logon name that is allowed access to this data. The PASSWORD option is required only if the USERNAME option is used.

If you specify USERNAME and PASSWORD in the **WEB SEND** command and you also specify AUTHENTICATE in the URIMAP resource, the WEB SEND values are used.If the specified password is over 8 characters long, it is treated as a password phrase when sent to z/OS systems.

PASSWORDLEN*(data-value)*
Specifies the buffer length supplied for the PASSWORD option as a fullword binary variable.

PATH*(data-area)*
Specifies the path information for the specific resource in the server that the application needs to access.

If the URIMAP option was used to specify an existing URIMAP definition on the WEB OPEN command for this connection, the path specified in that URIMAP definition is the default path for the WEB SEND command. In these circumstances, if you do not specify path information on the WEB SEND command, the path from the URIMAP definition is used. If you specify a different path from that given in the URIMAP definition, that path overrides the path from the URIMAP definition.

If the URIMAP option was not used on the WEB OPEN command, there is no default path, and you must provide path information. You can extract path information from a known URL using the WEB PARSE URL command.

As an alternative to using the PATH option to provide the path information, you can use the URIMAP option on the WEB SEND command to specify a URIMAP definition from which the path information is taken directly.

PATHLENGTH*(data-value)*
Specifies the length of the path, as a fullword binary value. If you are providing path information using the PATH option, you must specify the PATHLENGTH option. Path length information is returned if you use the WEB PARSE URL command to parse a URL.

QUERYSTRING*(data-area)*
Specifies a query string that is to be supplied to the server as part of the request. You do not have to include a question mark (?) at the beginning of the query string; if you do not include it, CICS supplies it for you automatically when constructing the request. If you include escaped characters in the query string, CICS passes them to the server in their escaped format.

QUERYSTRLEN*(data-value)*
Specifies the length of the query string supplied on the QUERYSTRING option, as a fullword binary value.

SESSTOKEN*(data-value)*
Specifies the session token, an 8-byte binary value that uniquely identifies a connection between CICS and a server. This value is returned by a WEB OPEN command for CICS as an HTTP client. Session tokens explains the use of the session token.

URIMAP*(data-value)*
Specifies the name, up to 8 characters, in mixed case, of a URIMAP definition that provides the path information for the specific resource in the server that the application will access. The URIMAP definition must be for CICS as an HTTP client, with USAGE(CLIENT) specified. Its HOST attribute must be the same as the HOST attribute of the URIMAP definition that was specified on the WEB OPEN command for this connection, or the same as the host name specified in the HOST option on the WEB OPEN command for this connection. A URIMAP definition specified on the WEB SEND command applies only to this request.

If the URIMAP option is specified, do not specify the PATH or PATHLENGTH options.

USERNAME*(data-value)*
Specifies the user ID or logon name that is allowed access to this data. If the USERNAME is specified, you must also use the PASSWORD option.

If you specify USERNAME and PASSWORD in the **WEB SEND** command and you also specify AUTHENTICATE in the URIMAP resource, the WEB SEND values are used.

USERNAMELEN*(data-value)*
Specifies the buffer length supplied for the USERNAME option as a fullword binary variable.

Conditions

122 CHANNELERR
RESP2 values are:

2 The channel specified by the CHANNEL option was not found.

110 CONTAINERERR
RESP2 values are:

2 The container specified by the CONTAINER option was not found.

16 INVREQ
RESP2 values are:

11 Action code invalid.

12 URIMAP and PATH are both specified. Only one is allowed. Or, the URIMAP option is not allowed for second or subsequent chunks.

13 Close status invalid.

15 Code page conversion failure.

17 Expect-100 request was rejected by the server.

22 Invalid chunk size.

32 Media type invalid.

33 Method does not support a body.

34 Method requires a body.

43 The DOCSTATUS value specified is invalid.

45 The character set specified is invalid.

46 The CLIENTCONV option is invalid.

49 The format of the path option is invalid.

54 The HTTP method is not valid.

63 URIMAP object disabled.

64 Host in URIMAP definition does not match the host specified when this session was opened.

69 Chunked transfer-coding not supported with this HTTP version.

71 Chunked transfer-coding error.

74 The connection has been closed. The server might have timed out because of inactivity on this connection.

76 MEDIATYPE option required.

79 Pipelining is in progress. Expect header cannot be sent.

80 CHARACTERSET cannot be specified with NOCLICONVERT.

120 The CHUNKING option is invalid.

121 FROMLENGTH option required.

122 FROM option required.

123 No message body specified. Use FROM, DOCTOKEN, or CHUNKEND.

124 CHUNKING option not specified, FROMLENGTH option required.

125 CHUNKNO specified, FROM option required.

126 CHUNKNO specified, FROMLENGTH option required.

127 CHUNKYES specified, FROM option required.

128 CHUNKYES specified, FROMLENGTH option required.

129 FROM option not allowed with CHUNKEND.

130 FROMLENGTH option not allowed with CHUNKEND.

131 FROMLENGTH option specified as zero.

132 METHOD option not allowed for second or subsequent chunks.

133 MEDIATYPE option not allowed for second or subsequent chunks.

135 PATH option not allowed for second or subsequent chunks.

136 METHOD option required.

142 AUTHENTICATE is invalid. The CVDA is not NONE or BASICAUTH.

144 One or more of the Web command parameters is invalid.

145 Channel was not specified, and there is no current channel.

147 Internal conversion error.

150 Conversion requested, but the data to be sent is in a DATATYPE BIT container.

17 IOERR

RESP2 values are:

42 Socket error.

22 LENGERR

RESP2 values are:

5 The PATHLENGTH option value was not greater than zero.

8 The QUERYSTRLEN option value was not greater than zero.

50 The FROMLENGTH option value was not greater than zero.

139 USERNAMELEN is negative or is greater than 256.

140 PASSWORDLEN is negative or is greater than 256.

70 NOTAUTH

RESP2 values are:

100 Path barred by security exit.

110 XWBAUTH error. The XWBAUTH global user exit has issued a UERCERR return code because the XWBAUTH exit is required but cannot return a valid response.

This error code is issued when the following are true: BASICAUTH is specified; USERNAME, PASSWORD, or both are omitted; XWBAUTH is inactive or returns a response of UERCERR.

13 NOTFND

RESP2 values are:

61 The URIMAP object specified was not found.

19 NOTOPEN
RESP2 values are:

27 Invalid session token.

124 TIMEDOUT

156 Timeout on socket send.

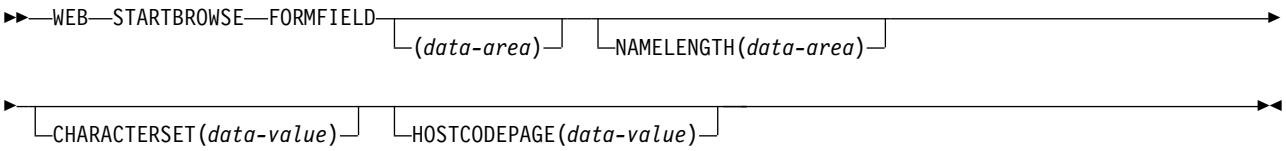
112 TOKENERR
RESP2 values are:

47 The document token specified is invalid or the document has been deleted.

WEB STARTBROWSE FORMFIELD

Signal start of HTML form field browse.

WEB STARTBROWSE FORMFIELD



Conditions: INVREQ, LENGERR, NOTFND

This command is threadsafe.

Description

WEB STARTBROWSE FORMFIELD signals the start of a browse of a set of name and value pairs in an HTML form that is part of the body of an HTTP request being processed by the current CICS task.

Options

CHARACTERSET(name)
specifies the 40–character name of the character set that is required for encoding the form data. This option should match the forms encoding determined by the corresponding HTML form (see How the client encoding is determined for more information). CICS does not support all the character sets named by IANA. HTML coded character sets lists the IANA character sets that are supported by CICS for code page conversion.

CLNTCODEPAGE(name)
This option is supported for upgrade purposes only. CHARACTERSET replaces it. The action taken by CICS is the same for both keywords.

FORMFIELD(data-area)
is the keyword that initiates the STARTBROWSE FORMFIELD command. You can optionally specify the name of the form field at which browsing is to start, by specifying this in a data-area, followed by the NAMELENGTH option, for example,

```
WEB STARTBROWSE FORMFIELD(name) NAMELENGTH(len)
```

The name is a string of text containing the name of the requested field. If a name is not specified, browsing starts at the first name and value pair in the HTML form.

HOSTCODEPAGE(name)
specifies the 8–character name of the CICS (host) code page required by the application program, into which the form data is to be converted. This code page is normally an EBCDIC code page.

The standard CICS form of a host code page name consists of the code page number (or more generally CCSID) written using 3 to 5 decimal digits as necessary then padded with trailing spaces to 8 characters. For code page 37, which is fewer than 3 digits, the standard form is 037. CICS now also accepts any decimal number of up to 8 digits (padded with trailing spaces) in the range 1 to 65535 as a code page name, even if it is not in the standard form.

If the code page is not specified, the data is returned in the EBCDIC code page specified by the LOCALCCSID system initialization parameter (which applies to the local CICS region, and has a default of 037), provided that the specified code page is supported by the CICS web interface. The code page is supported if it is one of a list of EBCDIC code pages that are recognized by CICS as being sufficiently standard to allow successful parsing of the web headers (this includes all SBCS CECP and Euro code pages). Otherwise, CICS returns the data in the default EBCDIC code page 037 instead.

NAMELENGTH(data-value)
specifies the length, as a fullword binary value, of the form field name. This field must be specified if a name data-area is specified with the FORMFIELD option.

Conditions

21 ILLOGIC
RESP2 value is:

5 A browse of form fields is already in progress.

16 INVREQ
occurs for the following conditions. RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request.

11 The client code page cannot be found.

12 The host code page cannot be found.

13 No forms data has been supplied in the HTTP request.

14 The code page combination for client and server is invalid.

17 Invalid forms data was found in the input message.

153 The form type is unknown.

154 A boundary string was expected in the forms data, but was not found.

22 LENGERR
occurs for the following conditions. RESP2 values are:

1 NAMELENGTH or VALUELENGTH is less than or equal to zero.

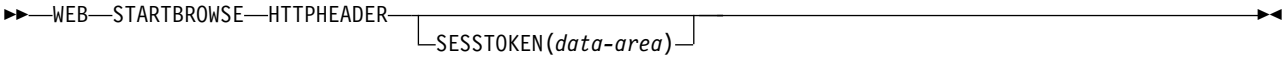
13 NOTFND
occurs for the following conditions. RESP2 values are:

1 The form field name given in the FORMFIELD parameter could not be found.

WEB STARTBROWSE HTTPHEADER

Signal start of HTTP header browse.

WEB STARTBROWSE HTTPHEADER



Conditions:ILLOGIC, INVREQ, NOTFND, NOTOPEN

This command is threadsafe.

Description

WEB STARTBROWSE HTTPHEADER signals the start of a browse of the HTTP header information. The SESSTOKEN option is required if the HTTP header information is part of a response sent to CICS as an HTTP client.

Options

SESSTOKEN*(data-value)*
For CICS as an HTTP client, this option is required. It specifies the session token, an 8-byte binary value that uniquely identifies a connection between CICS and a server. This value is returned by a WEB OPEN command for CICS as an HTTP client. Session tokens explains the use of the session token.

Conditions

21 ILLOGIC
RESP2 value is:

10 An HTTP header browse is already in progress.

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request.

43 No HTTP headers found.

13 NOTFND
RESP2 value is:

1 Header not found.

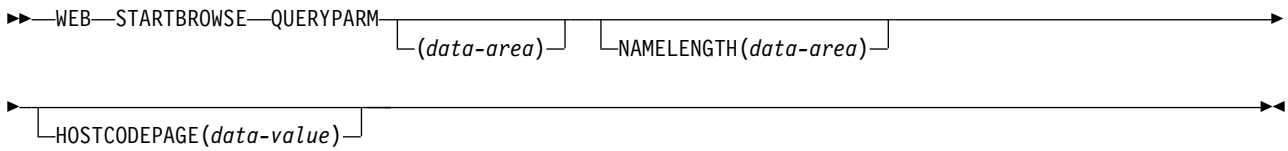
19 NOTOPEN
RESP2 value is:

27 Invalid session token.

WEB STARTBROWSE QUERYPARM

Start browsing query string data in a URL.

WEB STARTBROWSE QUERYPARM



Conditions: ILLOGIC, INVREQ, LENGERR, NOTFND

This command is threadsafe.

Description

WEB STARTBROWSE QUERYPARM signals the start of a browse of the keyword parameters, consisting of name and value pairs, from a query string in a URL.

For forms, you can also use the WEB STARTBROWSE FORMFIELD command. Forms are messages with the media types application/x-www-form-urlencoded or multipart/form-data.

Options

QUERYPARM(*data-area*)
is the keyword that initiates the STARTBROWSE QUERYPARM command. You can optionally specify the name of the keyword parameter at which browsing is to start, by specifying this in a data-area, followed by the NAMELENGTH option, for example,

```
WEB STARTBROWSE QUERYPARM(name) NAMELENGTH(len)
```

The name is a string of text containing the name of the requested keyword parameter. If a name is not specified, browsing starts at the first name and value pair in the query string.

HOSTCODEPAGE(*data-value*)
specifies the 8–character name of the CICS (host) code page required by the application program, into which the query string data is to be converted. This code page is normally an EBCDIC code page.

The standard CICS form of a host code page name consists of the code page number (or more generally CCSID) written using 3 to 5 decimal digits as necessary then padded with trailing spaces to 8 characters. For code page 37, which is fewer than 3 digits, the standard form is 037. CICS now also accepts any decimal number of up to 8 digits (padded with trailing spaces) in the range 1 to 65535 as a code page name, even if it is not in the standard form.

If the code page is not specified, the data is returned in the EBCDIC code page specified by the LOCALCCSID system initialization parameter (which applies to the local CICS region, and has a default of 037), provided that the specified code page is supported by the CICS web interface. The code page is supported if it is one of a list of EBCDIC code pages that are recognized by CICS as being sufficiently standard to allow successful parsing of the web headers (this

includes all SBCS CECP and Euro code pages). Otherwise, CICS returns the data in the default EBCDIC code page 037 instead.

NAMELENGTH(*data-value*)
specifies the length, as a fullword binary value, of the keyword parameter name. This field must be specified if a name data-area is specified with the QUERYPARM option.

Conditions

21 ILLOGIC
RESP2 value is:

5 A browse of keyword parameters is already in progress.

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

3 The command is being issued for a non-HTTP request.

12 The host code page cannot be found.

13 No keyword parameters found.

14 The code page combination for client and server is invalid.

17 Invalid keyword parameters found in the HTTP request.

22 LENGERR
RESP2 value is:

1 An invalid value for the NAMELEN parameter has been provided.

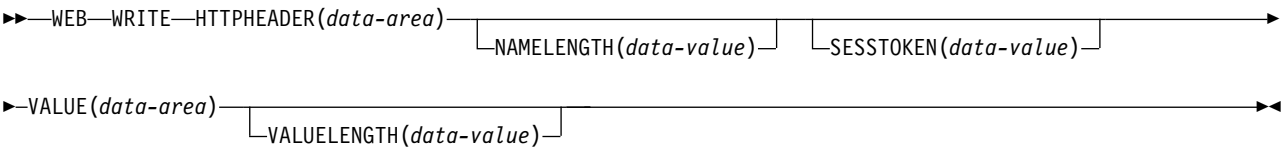
13 NOTFND
RESP2 value is:

1 Keyword parameter not found.

WEB WRITE HTTPHEADER

Build HTTP header information.

WEB WRITE HTTPHEADER



Conditions: INVREQ, LENGERR, NOTOPEN

This command is threadsafe.

Description

WEB WRITE HTTPHEADER enables an application to add HTTP header information to a message. When CICS is an HTTP server, the message is a response to a Web client. When CICS is an HTTP client, the message is a request to a server, and the SESSTOKEN option is specified.

Some HTTP headers are created automatically by CICS if the message requires them, and the application does not need to write these headers. These are:

- ARM correlator
- Connection
- Content-Type (written by CICS, but can be supplied by a client application if a complex header is required)
- Content-Length
- Date
- Expect
- Host
- Server (automatic creation depends on system initialization parameter **HTTPSERVERHDR**)
- TE (written by CICS but further instances may be added)
- Transfer-Encoding
- User-Agent (automatic creation depends on system initialization parameter **HTTPUSRAGENTHDR**)
- WWW-Authenticate

HTTP header reference for CICS web support describes the circumstances in which these headers are created. If the user application program writes a header that CICS also generates, CICS handles this depending on the situation:

- For CICS as an HTTP server, if the header is appropriate for a response, CICS does not overwrite it, but allows the application's version to be used.
- For CICS as an HTTP client, if the header is appropriate for a request, CICS does not allow the application to write it, and returns an error response to the **WEB WRITE HTTPHEADER** command. The exceptions are the TE header and the Content-Type header. Application programs can add further instances of the TE header. They can also supply the Content-Type header, if the required header needs to contain spaces or more than 56 characters, and so cannot be specified on the MEDIATYPE option of the WEB SEND command.
- If the header is not normally appropriate for the type of message (request or response), CICS allows it, as is the case for all user-defined headers. This situation should not occur if your message is compliant with the HTTP specification to which you are working.
- If the **WEB WRITE HTTPHEADER** is used to write a server header then a server header will be provided, even if the CICS supplied server header is suppressed by the **HTTPSERVERHDR=NO** parameter.

The **WEB WRITE HTTPHEADER** command adds a single header, and you can repeat the command to add further headers. If you write a header that you have already written for the request or response, CICS adds the new header to the request or response in addition to the existing header.

The name and value of the headers you write, and the circumstances in which you choose to write them, should conform to the requirements of the HTTP specification to which you are working.

If you want to use a header to request an action that might not be carried out correctly by a server or client below HTTP/1.1 level, and you need to confirm whether the action will succeed, use the WEB EXTRACT command with the HTTPVERSION option to check the HTTP version of the server.

For CICS as an HTTP client, if you are writing a Trailer header (for use with a chunked message) on your first request to the server, and you did not specify the options HTTPVNUM and HTTPRNUM on the **WEB OPEN** command for the session, CICS makes a request with the OPTIONS method to check the HTTP version of the server. This additional request is only made for the Trailer header.

The **WEB WRITE HTTPHEADER** command cannot be used if the connection with the server or Web client has been closed by either party sending a Connection: close header on a request or response.

For guidance on the correct use of this command:

- When writing headers for an HTTP response sent by CICS as an HTTP server, see Writing HTTP headers for a response.
- When writing headers for an HTTP request sent by CICS as an HTTP client, see .
- When using chunked transfer-coding to send an HTTP request or response, see Sending an HTTP request or response with chunked transfer-coding. That topic explains the correct procedure for writing trailing headers for a chunked message.

Options

HTTPHEADER*(data-area)*
Specifies the name of the HTTP header to be added to the request or response. The name, which is a string of text, should conform to the standards in the HTTP specification to which you are working.

NAMELENGTH*(data-value)*
Specifies the length, as a fullword binary value, of the HTTP header name.

SESSTOKEN*(data-value)*
For CICS as an HTTP client, this option is required. It specifies the session token, an 8-byte binary value that uniquely identifies a connection between CICS and a server. This value is returned by a WEB OPEN command for CICS as an HTTP client. Session tokens explains the use of the session token.

VALUE*(data-area)*
Specifies the value of the named HTTP header. The value, which is a string of text, should conform to the standards in the HTTP specification to which you are working.

VALUELENGTH*(data-value)*
Specifies the length, as a fullword binary value, of the HTTP header value.

Conditions

16 INVREQ
RESP2 values are:

1 The command is being issued in a non-CICS Web support application.

6 Client did not send TE: trailers on request, so trailing headers cannot be used.

19 Header not allowed. Some request headers may only be generated by CICS.

44 Header not allowed as a trailing header (trailer).

69 Chunked transfer-coding not supported.

70 Trailer header has not been created, so trailing headers cannot be written.

71 Chunked transfer-coding error.

74 Previous send failed.

78 Too late to write trailing headers for this message.

22 LENGERR
RESP2 values are:

35 The length in NAMELENGTH is not greater than zero.

55 The length in VALUELENGTH is not greater than zero or greater than 32000.

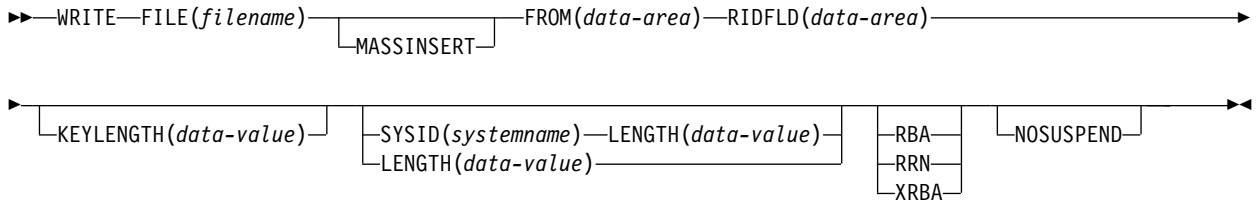
19 NOTOPEN
RESP2 values are:

27 Invalid session token.

WRITE

Write a record.

WRITE



Conditions: DISABLED, DUPREC, FILENOTFOUND, ILLOGIC, INVREQ, IOERR, ISCINVREQ, LENGERR, LOADING, LOCKED, NOSPACE, NOTAUTH, NOTOPEN, RECORDBUSY, SUPPRESSED, SYSIDERR

This command is threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over an IPIC connection to a remote CICS region.
- Defined as either local VSAM or RLS.

This command is not threadsafe if the file to which it refers is:

- Defined as remote and the command is function shipped over a non-IPIC connection.
- Defined as a shared data table, coupling facility data table, or BDAM file.

Description

WRITE writes a new record to a file on a local or a remote system.

When this command is used to write a record to a CICS-maintained data table, the update is made to both the source VSAM KSDS and the in-memory data table, unless the XDTAD user exit rejects the record from the table. The details of the command for a CICS-maintained table are the same as for a VSAM KSDS.

When this command is used to write a record to a user-maintained data table, the update is made to the in-memory data table (unless rejected by the XDTAD user exit).

When the command is used to write a record to a coupling facility data table, the update is made to the data table in the coupling facility (unless it is rejected by the XDTAD user exit).

If a write to a user-maintained table or coupling facility data table does not succeed because there is insufficient space, a NOSPACE response is returned to the command. CICS also issues message DFHFC0432 to alert system administrators to the situation.

For a VSAM ESDS or VSAM extended format, extended addressing ESDS, the record is always added at the end of the data set. CICS does not use the identification field specified in RIDFLD when calculating the relative byte address (RBA), or, for an extended addressing ESDS, the extended relative byte address (XRBA), of the new record. However, the new RBA or XRBA is returned to the application in the record identification field specified in the RIDFLD option.

For a VSAM KSDS, the record is added in the location specified by the associated key; this location may be anywhere in the data set. For VSAM data sets, the key in the record and the key in the RIDFLD identification field must be the same.

For a VSAM ESDS or KSDS, records can be either fixed-length or variable-length. MASSINSERT operations must proceed with ascending keys, and must be terminated by an UNLOCK before any other request to the same data set.

Options

FILE(*filename***)**
specifies the name of the file to be accessed.

If SYSID is specified, the data set to which this file refers is assumed to be on a remote system irrespective of whether the name is defined to CICS. Otherwise, the resource definition is used to find out whether the data set is on a local or a remote system.

FROM(*data-area***)**
specifies the record that is to be written to the data set referred to by this file.

KEYLENGTH(*data-value***)**
specifies the length (halfword binary) of the key that has been specified in the RIDFLD option, except when RBA or RRN is specified, in which case KEYLENGTH is not valid. You must code KEYLENGTH if you are also using SYSID (unless you are also using RBA or RRN). If the length specified is different from the length defined for the data set, the INVREQ condition occurs. The KEYLENGTH clause is required when the WRITE FILE is being function shipped, otherwise an INVREQ with RESP2=23 might occur.

LENGTH(*data-value***)**
specifies the length, as a halfword binary value, of the data area from which the record is written.

This option must be specified if SYSID is specified.

If the file is on a remote system and SYSID is *not* specified, the LENGTH parameter need not be set here but must be set in the file resource definition.

If the file is on a local system, the LENGTH option must be specified for variable-length records, but is optional for fixed-length records. It is, however, advisable to specify the length of fixed-length records because this causes CICS to check that the record being written is not longer than that defined for the data set.

If an incorrect length is specified for a WRITE to a file with fixed-length records, a record of the fixed length is written and the LENGERR condition is raised. If you supply too much data, the record is truncated. If you supply too little data, the record is padded with binary zeros.

MASSINSERT
(VSAM) specifies that the WRITE command is part of a mass-insert operation, that is, a series of WRITEs each specifying MASSINSERT.

See RLS Record level locking for information about using MASSINSERT on files opened in RLS access mode.

You cannot use MASSINSERT for user-maintained or coupling facility data tables.

NOSUSPEND (RLS only)
The request does not wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK.

A task could wait when it issues a WRITE request if the key is for a record that is being modified, created, or deleted by another task, because VSAM always acquires the lock first.

Note: Requests that specify NOSUSPEND wait for at least 1 second before CICS returns the RECORDBUSY response.

RBA
(VSAM ESDS base data sets only) specifies that the record identification field specified in the RIDFLD option contains a relative byte address. Use this option only when writing to an ESDS base.

RIDFLD(*data-area***)**
specifies the record identification field. The contents can be a key, a relative byte address, or relative record number (for VSAM data sets), or a block reference, a physical key, and a deblocking argument (for BDAM data sets). For a relative byte address or a relative record number, the format of this field must be fullword binary. If RBA or XRBA is specified, the RIDFLD is an output field that contains the relative byte address (greater than or equal to zero) of the record if the command is successful. This RBA value is calculated for you by CICS. If RRN is specified, RIDFLD contains the relative record number (greater than or equal to 1) of the record to be written.

See Identifying BDAM records and for more information about defining the record identification field.

When adding records to a keyed data set, the field must contain the complete key.

RRN
(VSAM RRDS) specifies that the record identification field specified in the RIDFLD option contains a relative record number.

SYSID(*systemname***)**
specifies the name of the system to which the request is directed.

If you specify SYSID and omit RBA, XRBA, and RRN, you must also specify LENGTH and KEYLENGTH; they cannot be found in the resource definition.

LENGTH must either be specified explicitly or must be capable of being defaulted from the FROM option using the length attribute reference in assembler language, or STG and CSTG in PL/I. LENGTH must be specified explicitly in C.

XRBA
specifies that the record identification field specified in the RIDFLD option contains an extended relative byte address. Use this option when writing to an extended addressing ESDS data set.

Conditions

84 DISABLED
RESP2 values:

50 A file was initially defined as disabled and has not since been enabled, or was disabled by a SET FILE or a CEMT SET FILE command.

Default action: terminate the task abnormally.

14 DUPREC
RESP2 values:

150 An attempt is made to add a record to a data set by referring to a file, or a path over a file (with the UNIQUEKEY attribute), in which the same key already exists.

This condition is also raised for a coupling facility data table that uses the contention model, even if another task has read the record with the same key for update. (For a coupling facility data table that uses the locking model, and for all other kinds of files, if another task has read the record for update, it is locked, and the WRITE request waits for the lock to be released, rather than returning a DUPREC response immediately.)

Default action: terminate the task abnormally.

12 FILENOTFOUND
RESP2 values:

1 A file name referred to in the FILE option is not defined to CICS.

Default action: terminate the task abnormally.

21 ILLOGIC
RESP2 values: (VSAM)

110 A VSAM error occurs that is not in one of the other CICS response categories.

See EIBRCODE in the EXEC interface block; for details, see EIB fields.

Default action: terminate the task abnormally.

16 INVREQ

RESP2 values:

20 Add operations are not allowed according to the resource definition.

23 When writing records containing embedded keys, the key in the record area (FROM option) and the key in RIDFLD do not match.

26 The KEYLENGTH option is specified, and the specified length does not equal the length defined for the data set that this file refers to.

38 A WRITE with the MASSINSERT option is issued against a BDAM file.

40 A BDAM key conversion error occurred.

44 The WRITE command does not conform to the format of WRITE for a user-maintained or coupling facility data table (for example, MASSINSERT or RBA is specified).

51 A WRITE command specifying the RBA keyword was issued against a KSDS file that is being accessed in RLS mode. RLS mode does not support relative byte address access to KSDS files.

55 NOSUSPEND is not allowed because the file is not a VSAM file that is accessed in RLS mode.

56 An attempt to update a recoverable coupling facility data table has failed because the current unit of work has already updated 1024 recoverable coupling facility data tables. You cannot update more than 1024 recoverable coupling facility data tables within a unit of work

57 WRITE operations are not allowed because the associated VSAM data set has an AVAILABILITY state of RREPL, which disallows such operations except those from replication programs.

59 XRBA was specified, but the data set is not an extended addressing ESDS.

Default action: terminate the task abnormally.

17 IOERR

RESP2 values:

120 There is an I/O error during the file control operation. An I/O error is any unusual event that is not covered by a CICS condition. Further information is available in the EXEC interface block; for details, see EIB fields.

For VSAM files, IOERR usually indicates a hardware error.

For BDAM files, IOERR could mean that you are trying to write to a BDAM track address that is not defined for the data set.

For a coupling facility data table, an IOERR indicates a bad response returned from a coupling facility access.

Default action: terminate the task abnormally.

54 ISCINVREQ

RESP2 values:

70 The remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

22 LENGERR

RESP2 values:

12 The length specified for the write operation exceeds the maximum record size; the record is truncated.

10 The LENGTH option is not specified. LENGTH must be specified for a WRITE to a file with variable-length records or to a BDAM file with records of undefined format.

14 An incorrect length is specified for a WRITE to a file with fixed-length records. A record of the fixed length has been written. If you supplied too much data, the record is truncated. If you supplied too little data, the record is padded with binary zeros.

94 LOADING

RESP2 values:

104 The request cannot be satisfied because it is issued against a data table that is still being loaded. The condition can be raised for one of the following reasons:

- The WRITE specifies a record key that has out of range of the records so far loaded into a coupling facility data table. Records can be added while a CFDT is loading only if the specified key is within the range of those records already loaded.

  The LOADING response can also be returned for a coupling facility data table that has failed during loading. For more information about what happens if the load for a coupling facility data table fails, see the description of the XDTLC global user exit in Data tables management exits (XDTRD, XDTAD, and XDTLC).
- A WRITE is issued to a user-maintained data table that is currently being loaded. A user-maintained data table cannot be modified during loading.

If your application programs encounter the LOADING condition persistently or too frequently, check that this is not caused by conflicting file definitions that reference the same data set.

Default action: terminate the task abnormally.

100 LOCKED

RESP2 values:

106 An attempt has been made to write a record, but a retained lock exists against the key of this record.

Default action: abend the task with code AEX8.

18 NOSPACE

RESP2 values:

100 No space is available on the direct access device for adding records to a data set.

102 The maximum number of table entries specified for the user-maintained table or coupling facility data table has already been reached.

This condition can also occur for a recoverable coupling facility data table when the table apparently contains fewer than the maximum number of records allowed if there are uncommitted updates outstanding.

103 CICS is unable to get sufficient storage in the CICS address space to create an in-memory table entry for the record being written.

108 There is insufficient space in the coupling facility data table pool to store the record.

Default action: terminate the task abnormally.

70 NOTAUTH

RESP2 values:

101 A resource security check has failed on FILE(filename).

Default action: terminate the task abnormally.

19 NOTOPEN

RESP2 values:

60 NOTOPEN (RESP2 60) is returned for one of the following reasons:

- The requested file is CLOSED and UNENABLED. The CLOSED, UNENABLED state is reached after a CLOSE request has been received against an OPEN ENABLED file and the file is no longer in use. You can also make CLOSED, UNENABLED the initial state, by specifying STATUS(UNENABLED) and OPENTIME(FIRSTREF) on the FILE resource definition. (For BDAM files, you use the FILSTAT parameter of the DFHFCT TYPE=FILE macro.)
- The requested file is OPEN and in use by other transactions, but a CLOSE request against the file has been received.
- A WRITE request is issued against a data set is quiesced, or is being quiesced, as a result of a SET DSNAME QUIESCED or IMMQUIESCED command.
- The requested file is CLOSED and ENABLED, so CICS has tried to open the file as part of executing the request. This file open has failed for some reason. You should examine the console for messages that explain why the file open has been unsuccessful.

This condition does not occur if the request is made to a CLOSED, DISABLED file. In this case, the DISABLED condition occurs.

Default action: terminate the task abnormally.

101 RECORDBUSY

RESP2 values:

107 NOSUSPEND is specified on the request but VSAM holds an active lock against the record, which would cause the request to wait. See Retained and active locks for more information.

Default action: abend the task with code AEX9.

72 SUPPRESSED

RESP2 values:

105 A user exit program that is invoked at the XDTAD exit point decides not to add the record to the user-maintained or coupling facility data table.

Default action: terminate the task abnormally.

53 SYSIDERR

RESP2 values:

130 The SYSID option specifies a name that is neither the local CICS region

nor a remote system defined to CICS by a CONNECTION definition. SYSIDERR also occurs when the link to the remote system is closed.

131 For a coupling facility data table, the connection to the coupling facility data table server has failed. This could be because the server itself has failed, or the server is available, but CICS has failed to connect to it.

132 The WRITE is issued against a coupling facility data table that no longer exists, probably because of a coupling facility failure, in which case the coupling facility data table server also fails. See Setting up and running a coupling facility data table server for information about restarting a coupling facility data table server and reloading a table.

Default action: terminate the task abnormally.

Retained and active locks: RECORDBUSY refers to active locks, and LOCKED refers to retained locks:

- READNEXT requests for records that have *retained* locks are always rejected with a LOCKED response.
- READNEXT requests for records that have *active* locks wait for the lock to be released, except when the NOSUSPEND keyword is specified, in which case CICS returns the RECORDBUSY response.

Examples

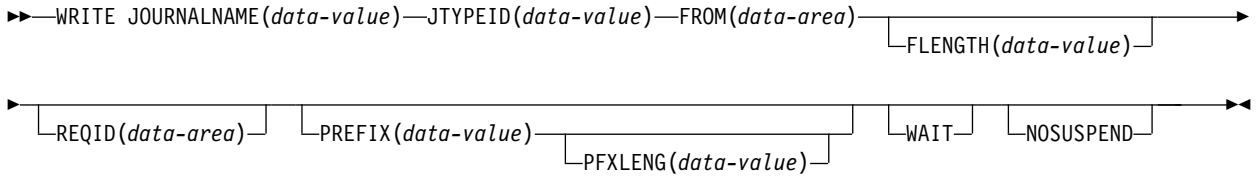
Here is an example of a simple WRITE command:

```
EXEC CICS WRITE
     FROM(RECORD)
     LENGTH(DATLEN)
     FILE('MASTER')
     RIDFLD(KEYFLD)
```

WRITE JOURNALNAME

Create a journal record

WRITE JOURNALNAME



Conditions: INVREQ, IOERR, JIDERR, LENGERR, NOJBUFSP, NOTAUTH, NOTOPEN

This command is threadsafe.

Description

WRITE JOURNALNAME writes a journal record from the specified data area to the system logger log stream that corresponds to the CICS journal name, or to SMF. The request can be for synchronous or asynchronous output; definitions of these terms, and information regarding the synchronization of journal output, are in Journal output synchronization .

Options

FLENGTH(*data-value***)**
specifies, as a full word binary value, the length in bytes of the user data to be built into the journal record.

Note that the maximum total length of a journal record depends on a number of factors:

- There is a limit of 32KB minus 400 bytes if the journal is using SMF.
- The limit for journals that map to log streams is the value expressed in the MAXBUFSIZE attribute for the structure being used minus 400 bytes. This has to include the user data, the prefix data, and the 2-byte JTYPEID.

Note: Data longer than 32K bytes cannot be read by offline jobs using the SUBSYS=LOGR interface.

FROM(*data-area***)**
specifies the user data to be built into the journal record.

JOURNALNAME(*data-value***)**
specifies a 1- to 8-character journal name. The valid characters for a journal name are the upper-case letters A through Z, the numeric characters 0 through 9, and the special symbols $ @ and #.

On first reference to this journal name, CICS must be able to map the journal name to a corresponding MVS system loggerlog stream, or MVS SMF data set. To do this, CICS searches the installed JOURNALMODEL definitions, looking for a matching journal name in a journal model. CICS looks for either a specific match or a generic match. If a matching entry cannot be found, CICS attempts to use a default log stream name.

To write to the CICS system log, specify DFHLOG as the journal name.

Note: The CICS system log should be used only for short-lived data required for recovery purposes. You should not write user records for such things as audit trails to it.

To write to journals defined using the journal numbering convention (for example, to the auto journals defined in file resource definitions), specify the name as DFHJ*nn*, where nn is the journal number in the range 1 to 99.

You cannot write to a forward recovery log that is known to CICS only by its 26-character log stream name (as derived directly from the VSAM ICF catalog) unless you write to a journal whose matching JOURNALMODEL is associated with the same log stream name.

Specifying DFHJ01 on this command refers to a user journal, *not* the system log.

JTYPEID(*data-value***)**
specifies a 2-character identifier to be placed in the journal record to identify its origin.

NOSUSPEND
specifies that the application program is not to be suspended for the NOJBUFSP condition. The user record is ignored.

PFXLENG(*data-value***)**
specifies the length (halfword binary value) in bytes of the user prefix data to be included in the journal record.

Note that the maximum total length of a journal record depends on a number of factors:

- There is a limit of 32KB minus 400 bytes if the journal is using SMF.
- The limit for journals that map to log streams is the value expressed in the MAXBUFSIZE attribute for the structure being used minus 400 bytes. This has to include the prefix data, the user data, and the 2-byte JTYPEID.

The minimum value is 0. See FLENGTH for the limits to the size of a journal record.

Note: Data longer than 32K bytes cannot be read by offline jobs using the SUBSYS=LOGR interface.

PREFIX(*data-value***)**
specifies the user prefix data to be included in the journal record. A data area must be provided in COBOL programs.

REQID(*data-area***)**
specifies a data area that identifies the journal record. The data area is a fullword binary variable. CICS sets the variable to a token that can be used for synchronization. REQID is only valid for asynchronous output (that is, the WAIT option is not specified).

WAIT
specifies that synchronous journal output is required. The requesting task waits until the record has been hardened.

Conditions

16 INVREQ
the command is not valid for processing by CICS.

Default action: Terminate the task abnormally.

17 IOERR
a journal record has not been output because an irrecoverable error condition was returned by the system logger log stream or by SMF.

Default action: If the log is the system log, CICS either quiesces or abends CICS. If the log is a general log, the task is terminated abnormally.

43 JIDERR
CICS cannot connect to a log stream referenced by the specified journal name, for one of the following reasons:

- The log stream does not exist and cannot be created dynamically using default model definitions.
- The log stream is a DASD-only log stream to which a CICS region in another MVS image is currently connected.

Default action: terminate the task abnormally.

22 LENGERR
the aggregate length of the journal record, comprising the user data (FROM,

JTYPE, and PREFIX data) and the CICS header data, is too great to fit the maximum block size allowed for the log stream.

Default action: terminate the task abnormally.

45 NOJBUFSP
the journal buffers are logically full (that is, the current buffer has insufficient space for this journal record, and I/O is in progress on the alternate buffer).

Default action: CICS suspends task activity until the journal request can be satisfied. CICS ensures that both buffers are written out to auxiliary storage, thus freeing them for new records. (You can override the default action by the NOSUSPEND option.)

70 NOTAUTH
a resource security check has failed on the JOURNALNAME(data-value).

Default action: terminate the task abnormally.

19 NOTOPEN
occurs in any of the following situations:

- The command cannot be executed because the specified journal has been explicitly disabled by the user.
- The request could not be satisfied because the specified journal was defined using a journal model that maps it onto the logstream that is being used as the system log for this CICS system. The error is detected when trying to connect to the logstream and results in a definition for the JOURNALNAME being installed and set to 'failed'.

Default action: terminate the task abnormally.

Examples

The following example shows how to write synchronous journal output and wait for the output operation to be completed:

```
EXEC CICS WRITE
     JOURNALNAME('ACCTSJNL')
     JTYPEID('XX')
     FROM(KEYDATA)
     FLENGTH(40000)
     PREFIX(PROGNAME)
     PFXLENG(6)
     WAIT
```

The following example shows how to write deferred (asynchronous) user recovery data to the CICS system log:

```
EXEC CICS WRITE
     JOURNALNAME('DFHLOG')
     JTYPEID('UR')
     FROM(COMDATA)
     FLENGTH(10)
     REQID(ENTRYID)
```

WRITE JOURNALNUM

Create a journal record.

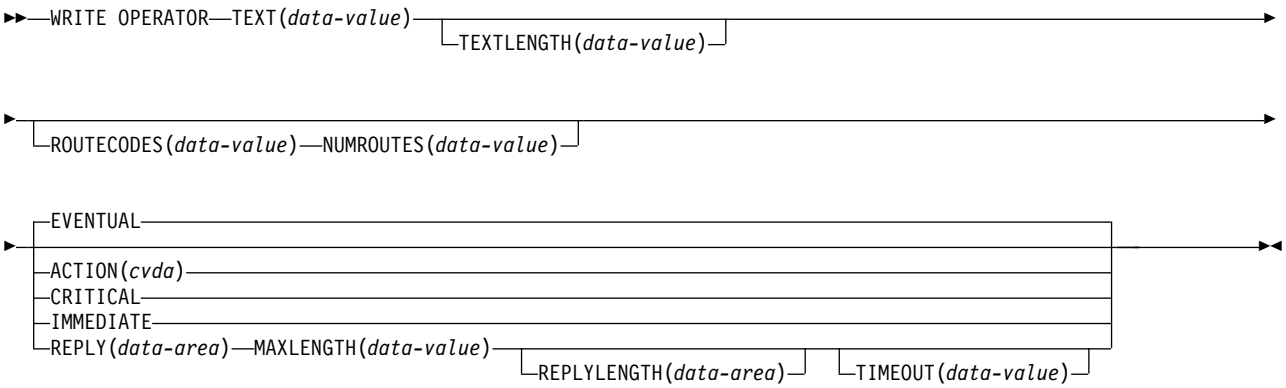
This command is supported for compatibility with earlier releases of CICS. It is superseded by the WRITE JOURNALNAME command, which you are recommended to use instead.

This command is threadsafe.

WRITE OPERATOR

Write a message on the system console.

WRITE OPERATOR



Conditions: EXPIRED, INVREQ, LENGERR

This command is threadsafe.

Note: This command is threadsafe, but the REPLY option is not compatible with a key 9 TCB. If an **EXEC CICS WRITE OPERATOR** command specifying the REPLY option is issued from a key 9 TCB then CICS will switch to the QR TCB to process the command.

Description

WRITE OPERATOR enables an application to write a message to one or more system consoles and, if required, wait for a reply. The command can specify route codes, which is of particular use to application packages that have to issue their own operator messages.

As a result of a change in the way CICS handles messages sent to the console, text lengths of greater than 113 characters are split into two lines. None of the variables below can be defined as PL/I variable character strings.

Note: If ACTION (or one of the equivalent CVDA values below) is specified, the message is retained until the console operator explicitly deletes it or CICS terminates.

The action code is identical with the descriptor code to be associated with the message. Only one of the descriptor codes 2, 3, or 11 may be specified for this parameter.

If ACTION is not specified, no descriptor code is associated with the message. The descriptor codes have the following meanings:

2 Immediate action

3 Eventual action

11 Critical eventual action.

The CRITICAL option is equivalent to a specification of ACTION(11). The EVENTUAL option is equivalent to a specification of ACTION(3). The IMMEDIATE option is equivalent to a specification of ACTION(2).

Retained messages can be handled by the console operator in a variety of ways (see *z/OS MVS System Commands*). Refer to your system programmer for information about how this command affects the appearance of the console screen to the operator.

Options

ACTION(*cvda***)**
Specifies an action code that is associated with this message. CVDA values are:

CRITICAL
Specifies that the message requires eventual action by the operator and has enough critical importance to remain on the console screen. The message remains on the screen until it is deleted by the operator.

EVENTUAL
Specifies that the operator should take action when there is time. The message is rolled off when other messages fill up the screen, but is still retained by the operating system until the operator explicitly deletes it.

IMMEDIATE
Specifies that the operator should take action immediately. The message remains on the console screen until it is deleted by the operator.

MAXLENGTH(*data-value***)**
Specifies a fullword binary field that contains the length of the reply area (in the range 1–119 bytes). You must specify MAXLENGTH if you specify REPLY.

NUMROUTES(*data-value***)**
Specifies a fullword binary field that defines the number of routing codes.

REPLY(*data-area***)**
Specifies a data area for receiving the operator's reply. If you specify this option, your application pauses until either a reply is received or the TIMEOUT period expires.

REPLYLENGTH(*data-area***)**
Specifies the actual length (fullword binary value) of the operator's reply.

ROUTECODES(*data-value***)**
Specifies a variable-length field. Each code is one byte and contains a binary number in the range 1–28. The default is a single code, set to 2. In COBOL programs only, you must use a data-area that contains the 1-byte values rather than a data-value.

TEXT(*data-value***)**
Specifies a data value containing the text to be sent.

If the data value begins with DFHnnnn or DFHaannnn, the message is treated as a CICS message and is reformatted accordingly.

If you are using the COBOL2 translator option, you must use a data-area that contains the text to be sent to the operator, and not a data-value.

TEXTLENGTH(*data-value***)**
Specifies the length, as a fullword binary value, of the text. This option is required only for C and C++ programs.

- If the REPLY option is specified, the length is in the range 0–121 bytes.
- If the REPLY option is not specified, the length is in the range 0–690 bytes.

If the length of the text is greater than 113, CICS formats the message in a multiline write to operator (WTO); each line has 69 bytes with a maximum of ten lines.

The output is edited in such a way that each line is broken, if possible, at a space character, The next line starts with a non-space character. If there is no room to reformat the data within the overall limit of 690 bytes of ten lines of 69 bytes, the output is not reformatted.

TIMEOUT(*data-value***)**
Specifies a fullword binary field that contains the maximum time (in seconds) that CICS waits for a reply before returning control to this transaction. This must be in the range 0–86 400 (24 hours). The system default value is specified by the **OPERTIM** system initialization parameter. You can only specify TIMEOUT if you have also specified REPLY.

Conditions

31 EXPIRED
RESP2 values:

7 TIMEOUT has occurred before the operator's reply was received.

Default action: return the exception condition to the application.

16 INVREQ
RESP2 values:

1 The TEXTLENGTH value is not valid.

2 The NUMROUTES value is not valid.

3 The ROUTECODES value is not valid.

4 The MAXLENGTH value is not valid.

5 The TIMEOUT value is not valid.

6 The ACTION value is not valid.

Default action: terminate the task abnormally.

22 LENGERR
RESP2 values:

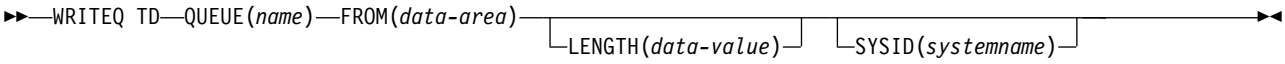
8 The reply was longer than MAXLENGTH, and has been truncated.

Default action: terminate the task abnormally.

WRITEQ TD

Write data to transient data queue.

WRITEQ TD



Conditions: DISABLED, ERROR, INVREQ, IOERR, ISCINVREQ, LENGERR, LOCKED, NOSPACE, NOTAUTH, NOTOPEN, QIDERR, SYSIDERR

This command is threadsafe when it is used with a queue in a local CICS region, or function shipped to a remote CICS region over an IPIC connection. It is non-threadsafe when it is function shipped to a remote CICS region over another type of connection.

Description

WRITEQ TD writes transient data to a predefined symbolic destination.

Options

FROM(*data-area***)**
Specifies the data that is to be written to the transient data queue.

LENGTH(*data-value***)**
Specifies the length (halfword binary value) of the data to be written.

QUEUE(*name***)**
Specifies the symbolic name (1 - 4 alphanumeric characters) of the queue to which the data is written. The named queue must have been defined to CICS.

SYSID(*systemname***)**
(remote systems only) Specifies the name (1 - 4 characters) of the system to which the request is directed.

If SYSID is specified, the queue is assumed to be on a remote system whether or not it is defined as remote. Otherwise the transient data queue definition is used to find out whether the data set is on a local or a remote system.

Conditions

84 DISABLED
Occurs when the queue has been disabled.

Default action: terminate the task abnormally.

1 ERROR
Occurs for an error that does not raise any other condition.

Default action: terminate the task abnormally.

16 INVREQ
Occurs if WRITEQ names an extrapartition queue that has been opened for input.

Note: This condition cannot be raised for intrapartition queues.

Default action: terminate the task abnormally.

17 IOERR
Occurs when an input/output error occurs and the data record in error is skipped.

Default action: terminate the task abnormally.

54 ISCINVREQ
Occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

22 LENGERR
Occurs in either of the following situations:

- WRITEQ names an extrapartition queue and does not specify a length consistent with the RECORDSIZE and associated formations specified in the TDQUEUE resource definition. The check is made after the XTDOUT exit has been invoked; this exit may change the length of the data to be passed to the access method.
- WRITEQ names an intrapartition queue and does not specify a length consistent with the control interval defined for the intrapartition data set. Again, the check is made after the XTDOUT exit has been invoked.

Default action: terminate the task abnormally.

100 LOCKED
Occurs when the request cannot be performed because use of the queue has been restricted owing to a unit of work failing indoubt. This can happen on any request for a logically-recoverable queue defined with WAIT(YES) and WAITACTION(REJECT) in the TDQUEUE resource definition.

Specify WAIT(YES) and WAITACTION(QUEUE) in the TDQUEUE resource definition if you want the transaction to wait.

Default action: terminate the task abnormally.

18 NOSPACE
Occurs if no more space exists on the intrapartition or extrapartition queue, or the relative byte address (RBA) for an intrapartition queue would exceed 2 GB. When this happens, no more data should be written to the queue because it may be lost.

Default action: terminate the task abnormally.

70 NOTAUTH
Occurs when a resource security check has failed on QUEUE(*name*).

Default action: terminate the task abnormally.

19 NOTOPEN
Occurs if the destination is closed.

Note: This condition cannot be raised for intrapartition queues.

Default action: terminate the task abnormally.

44 QIDERR
Occurs if the symbolic destination to be used with a transient data control command cannot be found.

Default action: terminate the task abnormally.

53 SYSIDERR
Occurs when the SYSID option specifies a name that is neither the local system

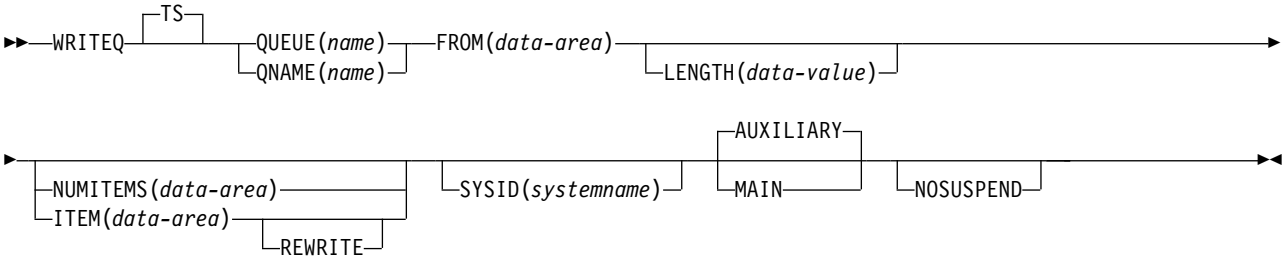
nor a remote system (made known to CICS by defining a CONNECTION or an IPCONN). SYSIDERR also occurs when the link to the remote system is closed.

Default action: terminate the task abnormally.

WRITEQ TS

Write data to a temporary storage queue.

WRITEQ TS



Conditions: INVREQ, IOERR, ISCINVREQ, ITEMERR, LENGERR, LOCKED, NOSPACE, NOTAUTH, QIDERR, SYSIDERR

This command is threadsafe when it is used with a queue in main storage or auxiliary storage, either in a local CICS region, or function shipped to a remote CICS region over an IPIC connection. It is also threadsafe when it is used with a queue in a shared temporary storage pool in a z/OS coupling facility that is managed by a temporary storage data sharing server (TS server). The command is non-threadsafe when it is function shipped to a remote CICS region over another type of connection other than IPIC.

Note for dynamic transaction routing: Using this command might create inter-transaction affinities that adversely affect the use of dynamic transaction routing. For more information about transaction affinities, see Affinity .

Description

WRITEQ TS stores temporary data records in a temporary storage queue in main or auxiliary storage.

If a queue is defined as recoverable, the program must not issue a **WRITEQ TS** command if a **DELETEQ TS** command has previously been issued in the same logical unit of work. That is, following a **DELETEQ TS** command, a **WRITEQ TS** command must not be issued until after a sync point occurs.

If there is insufficient space available in the temporary storage data set or main storage to satisfy the **WRITEQ TS** request, the task is suspended until space does become available. (Other tasks in the system might release space.) If space is not available and you specified the **NOSUSPEND** option, the NOSPACE condition is raised, and you can decide whether to stop the transaction with an abend, or wait.

Options

AUXILIARY
Specifies that the temporary storage queue is on a direct access storage device in auxiliary storage. This is the default value for the first write.

This option is ignored in the following situations:

- For an existing queue
- If a TSMODEL resource definition with a matching prefix is installed in the system
- If the specified temporary storage data queue resides in a shared temporary storage pool
- If CICS has been initialized with TS main-only support, that is, the system initialization parameter TS=(,0) has been specified

FROM(*data-area***)**
Specifies the data to be written to temporary storage.

ITEM(*data-area***)**
Specifies, as a halfword binary value, the item number of the logical record to be replaced in the queue (REWRITE option also specified).

ITEM can be both an input and output field to CICS. Therefore, programmers must ensure that the ITEM field is not defined in protected storage when issuing a WRITEQ command. If the ITEM value was a literal (for example), command checking (CMDPROT=YES) would result in an AEYD abend occurring.

Note: In earlier releases, ITEM on a WRITEQ TS without REWRITE would perform a similar function to NUMITEMS. This function is retained for compatibility.

LENGTH(*data-value***)**
Specifies the length, as a halfword binary value, of the data to be written.

You must specify this option if you are using SYSID.

The maximum length is 32763. For a description of a safe upper limit, see LENGTH options in CICS commands.

MAIN
Specifies that the temporary storage queue is in main storage.

This option is ignored in the following situations:

- For an existing queue
- If a TSMODEL resource definition with a matching prefix is installed in the system
- If the specified temporary storage data queue resides in a shared temporary storage pool

If you use the MAIN option to write data to a temporary storage queue on a remote system, the data is stored in main storage if the remote system is accessed by the CICS multiregion operation (MRO) facility or IPIC connectivity. If these conditions are not met, the data is stored in auxiliary storage.

If the system is MRO and MAIN is specified, the queue is not recoverable and SYNCPOINT ROLLBACK does not function.

NOSUSPEND
Specifies that if there is insufficient space in the temporary storage data set or in main storage to satisfy the WRITEQ TS request, the application program is not suspended. The NOSPACE condition is raised instead.

However, if a HANDLE CONDITION command for NOSPACE is active when the command is executed, this condition also overrides the default action, and control is passed to the user label supplied in the HANDLE CONDITION

command. This condition takes precedence over the NOSUSPEND option but is, of course, negated by either NOHANDLE or RESP.

NUMITEMS(*data-area***)**
Specifies a halfword binary field where CICS stores a number that indicates how many items there are now in the queue, after the WRITEQ TS command is executed.

If the record starts a new queue, the item number assigned is 1; subsequent item numbers follow on sequentially. NUMITEMS is not valid if REWRITE is specified.

QNAME(*name***)**
An alternative to QUEUE, QNAME specifies the symbolic name (1 - 16 characters) of the queue to be written to. If the name has less than 16 characters, you must still use a 16-character field, padded with blanks if necessary. If the queue is defined to CICS as remote, the request is shipped to a remote system. Do not use X'FA' through X'FF', or **, or $$, or DF, as the first character of the name; these characters are reserved for CICS use. The name cannot consist solely of binary zeros.

QUEUE(*name***)**
Specifies the symbolic name (1 - 8 characters) of the queue to be written to. If the name has less than 8 characters, you must still use an 8-character field, padded with blanks if necessary. If the queue is defined to CICS as remote, the request is shipped to a remote system. Do not use X'FA' through X'FF', or **, or $$, or DF, as the first character of the name; these characters are reserved for CICS use. The name cannot consist solely of binary zeros.

REWRITE
Specifies that the existing record in the queue is to be overwritten with the data provided. If the REWRITE option is specified, the ITEM option must also be specified. If the specified queue does not exist, the QIDERR condition occurs. If the correct item within an existing queue cannot be found, the ITEMERR condition occurs and the data is not stored.

SYSID(*systemname***)**
(Remote and shared queues only) Specifies the system name (1 - 4 characters) identifying the remote system or shared queue pool to which the request is directed. Note that TSMODEL resource definitions do not support specifying a SYSID for a queue that resides in a temporary storage data sharing pool. Use the QUEUE or QNAME option instead. Using an explicit SYSID for a shared queue pool requires the support of a temporary storage table (TST).

Conditions

16 INVREQ
Occurs in any of the following situations:

- A WRITEQ TS command specifies a queue name that consists solely of binary zeros.
- A WRITEQ TS command specifies a queue that is locked and awaiting ISC session recovery.
- The queue was created by CICS internal code.

Default action: terminate the task abnormally.

17 IOERR
RESP2 values:

5 There is an irrecoverable input/output error for a shared queue.

Default action: terminate the task abnormally.

54 ISCINVREQ
Occurs when the remote system indicates a failure that does not correspond to a known condition.

Default action: terminate the task abnormally.

26 ITEMERR
Occurs in any of the following situations:

- The item number specified in a WRITEQ TS command with the REWRITE option, is not valid (that is, it is outside the range of entry numbers assigned for the queue).
- The maximum number of items (32767) is exceeded.

Default action: terminate the task abnormally.

22 LENGERR
Occurs in any of the following situations:

- The length of the stored data is zero or negative.
- The length of the stored data is greater than 32763.

Default action: terminate the task abnormally.

100 LOCKED
RESP2 values:

0 The request cannot be performed because use of the queue has been restricted owing to a unit of work failing indoubt.

Default action: terminate the task abnormally.

18 NOSPACE
Occurs when the NOSUSPEND option is specified and there is no space for the data in the following:

- Main storage
- The auxiliary temporary storage data set
- The temporary storage pool list structure

This condition also occurs if there is no space and there is an active HANDLE CONDITION for NOSPACE.

Default action: ignore the condition.

70 NOTAUTH
RESP2 values:

101 A resource security check has failed on QUEUE(name).

Default action: terminate the task abnormally.

44 QIDERR
Occurs when the queue specified by a WRITEQ TS command with the REWRITE option cannot be found in any of the following:

- Main storage
- Auxiliary storage
- Temporary storage pool

Default action: terminate the task abnormally.

53 SYSIDERR
RESP2 values:

4 Occurs in any of the following situations:

- The SYSID option specifies a name that is not the local system or a remote system (made known to CICS by defining a CONNECTION or an IPCONN).
- When IPIC connectivity is used, the local system, the remote system, or both, are not CICS TS 4.2 or later regions.
- The link to the remote system is closed.
- The CICS region in which the temporary storage command is executed fails to connect to the TS server managing the TS pool that supports the referenced temporary storage queue. For example, this situation can occur if the CICS region is not authorized to access the temporary storage server.

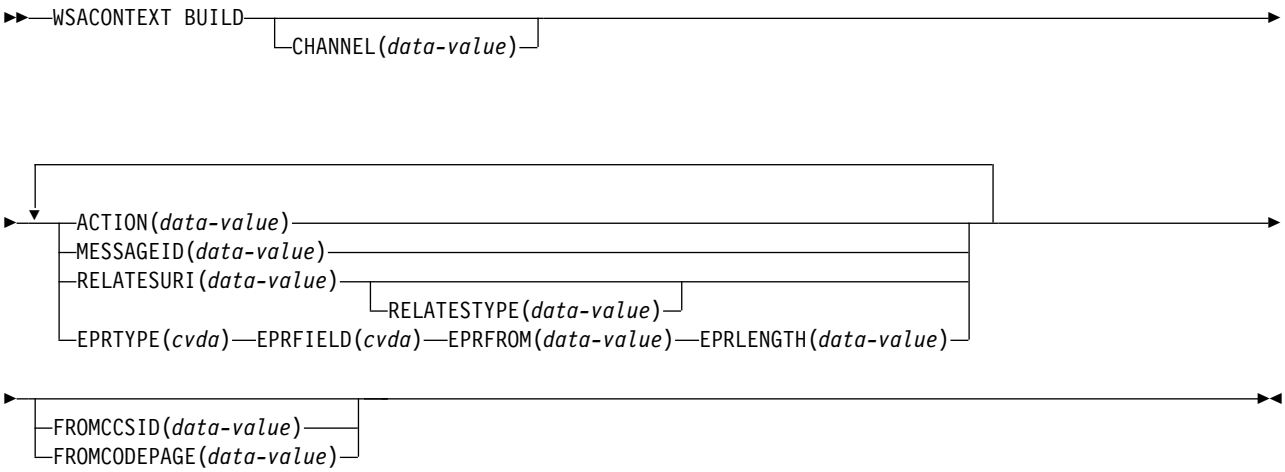
  This condition can also occur if the temporary storage server is not started, or because the server has failed (or been stopped) while CICS continues to run.

Default action: terminate the task abnormally.

WSACONTEXT BUILD

Use the **WSACONTEXT BUILD** command to insert or replace WS-Addressing message addressing properties (MAPs) in the addressing context.

WSACONTEXT BUILD



Conditions: CCSIDERR, CHANNELERR, CODEPAGEERR, INVREQ, LENGERR

This command is threadsafe.

Description

Use the **WSACONTEXT BUILD** command for any of the following actions:

- To insert or replace the Action or Message ID MAPs
- To insert or replace the To, From, ReplyTo, or FaultTo endpoint reference MAPs.
- To insert the RelatesTo MAPs.

You can use the command repeatedly to supply different data on each call, for example different endpoint references (EPRs). The MAPs are applied to all outbound SOAP messages created by the **INVOKE SERVICE** or **INVOKE WEBSERVICE** commands and to response SOAP messages from a service provider.

Options

ACTION(*data-value*)
Specifies an input value containing an Action MAP of the request or response SOAP message; for example, `http://example.ibm.com/namespace/bookingInterface/MakeBooking`. Actions are supplied in the WSDL or are calculated by the web services assistant, but can be overridden by this option. The data value must be 255 characters in length. If the Action MAP is less than 255 characters, you must pad the data value with trailing blanks.

CHANNEL(*data-value*)
Specifies the name of the channel that holds the addressing context. The name of the channel can be up to 16 characters in length. If the channel name is fewer than 16 characters, you must pad the data value with trailing blanks. If you do not specify this option, the current channel is implied.

Acceptable characters for the channel name are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blank characters are not permitted. The accepted set of characters for channel names includes some characters that do not have the same representation in all EBCDIC code pages. It is therefore recommended that, if channels are to be shipped between regions, the characters used in naming them should be restricted to A-Z a-z 0-9 & : = , ; < > . - and _.

You can specify the channel name `DFHTRANSACTION` to use the transaction channel.

This option is required when the **WSACONTEXT BUILD** command is used by a web service requester application and is the channel name used by subsequent **INVOKE SERVICE** commands.

EPRFIELD(*cvda*)
Specifies the endpoint reference field. You can use this option multiple times to build a full endpoint reference.

ADDRESS
The Address field of the endpoint reference is specified as a URI in the EPRFROM option.

ALL A complete endpoint reference, described in XML, is specified in the EPRFROM option.

METADATA
The Metadata section of the endpoint reference, described in XML, is specified in the EPRFROM option.

REFPARMS
The ReferenceParameters section of the endpoint reference, described in XML, is specified in the EPRFROM option.

EPRFROM(*data-value*)
An input data value that contains a complete or partial endpoint reference that is to be placed in the addressing context. The EPRFIELD option describes what part of the endpoint reference is specified in this option by the application. The endpoint reference can be supplied by a web service application that uses the **WSAEPR CREATE** command or from another source, such as a service registry.

If the EPRFROM option contains an address, any special characters in the address are automatically escaped or unescaped.

The following characters in an address are replaced with an escape sequence: ", >, <, ', &

EPRLENGTH(*data-value*)
A fullword binary input data value used to contain the length of the supplied EPR.

EPRTYPE(*cvda*)
Specifies the type of EPR that is being built:

TOEPR
The destination EPR to which a SOAP message is sent.

REPLYTOEPR
An EPR to which a SOAP response message is returned.

FAULTTOEPR
An EPR to which a SOAP fault message is returned.

FROMEPR
An EPR that represents the sender of the SOAP message.

FROMCCSID(*data-value*)
Specifies the current Coded Character Set Identifier (CCSID) of the character data to be put into the addressing context, as a fullword binary number. If you want to specify an IANA name for the code page or if you want to specify the CCSID as alphanumeric characters, use the FROMCODEPAGE option instead.

For CICS Transaction Server for z/OS applications, the CCSID is typically an EBCDIC CCSID.

The default CCSID of the region is specified on the **LOCALCCSID** system initialization parameter.

For an explanation of CCSIDs, and a list of the CCSIDs supported by CICS, see CICS-supported conversions.

FROMCODEPAGE(*data-value*)
Specifies an IANA-registered alphanumeric charset name or a Coded Character Set Identifier (CCSID) for the current code page of the character data to be put into the addressing context, using up to 40 alphanumeric characters including appropriate punctuation. Use this option instead of the CCSID option if you prefer to use an IANA-registered charset name, as specified in the Content-Type header for an HTTP request. CICS converts the IANA name into a CCSID, and the subsequent data conversion process is identical. Also use this option if you prefer to specify the CCSID in alphanumeric characters, rather than as a fullword binary number.

Where an IANA name exists for a code page and CICS supports its use, the name is listed with the CCSID in CICS-supported conversions.

MESSAGEID(*data-value*)
Specifies a data value for a URI that uniquely identifies a SOAP message. The data value must be 255 characters in length. If the URI is less than 255 characters, you must pad the data area with trailing blanks.

RELATESTYPE(*data-value*)
Specifies a URI denoting the relationship type between the message to be sent and another message. The value must be 255 characters long. If the URI is less

than 255 characters, you must pad the data value with trailing blanks. You are allowed to specify multiple RelatesTo MAPs. Subsequent calls of the RELATESTYPE and RELATESURI options create new RelatesTo MAPs.

If you do not specify a value for this option, the RelatesTo MAP does not have a type attribute in the SOAP message and defaults to a value of `http://www.w3.org/2005/08/addressing/reply`.

RELATESURI(*data-value*)
Specifies a URI denoting the message ID of a message that the message to be sent is related to. The value must be 255 characters long. If the URI is less than 255 characters long, you must pad the data value with trailing blanks.

Conditions

123 CCSIDERR
RESP2 values:

1 The CCSID specified on the FROMCCSID option is outside the range of valid CCSID values.

2 The CCSID specified on the FROMCCSID option and the CCSID of the addressing context are an unsupported combination.

4 One or more characters could not be converted. Each unconverted character has been replaced by a blank in the converted data.

5 There was an internal error in the code page conversion of the addressing context data.

6 Either the text encoding is not compatible with the specified CCSID on the FROMCCSID option, or one or more characters are truncated.

122 CHANNELERR
RESP2 values:

1 The name specified for the CHANNEL option contains an illegal character or combination of characters.

2 The specified channel was not located.

125 CODEPAGEERR
RESP2 values:

1 The code page specified on the FROMCODEPAGE option is not supported.

2 The code page specified on the FROMCODEPAGE option and the CCSID of the addressing context are an unsupported combination.

4 One or more characters could not be converted. Each unconverted character has been replaced by a blank in the converted data.

5 There was an internal error in the code page conversion of a container.

6 Either the text encoding is not compatible with the specified CCSID on the FROMCCSID option, or one or more characters are truncated.

16 INVREQ
The INVREQ RESP2 values and the corresponding messages are shown below. For this command, if the EIBRESP2 value is > 100, the fullword EIBRESP2 field is regarded as a structure containing two halfwords. The low-order halfword always contains the error number. The high-order halfword contains the offset into the XML data where the parsing error occurred.

RESP2 values:

4 The CHANNEL option was not specified. There is no current channel because the program that issued the command was not passed the name of a channel.

6 The ACTION field does not contain valid URI characters.

7 The MESSAGEID field does not contain valid URI characters.

8 The RELATESURI field does not contain valid URI characters.

9 The RELATESTYPE field does not contain valid URI characters.

10 The EPRFROM option does not contain valid XML.

13 The EPRFROM option does not contain valid XML. The EPR <Metadata> might not contain valid XML.

14 The EPRFROM option does not contain valid XML. The EPR <ReferenceParameters> might not contain valid XML.

15 The EPRFROM option might not contain a valid URI.

22 LENGERR

RESP2 values:

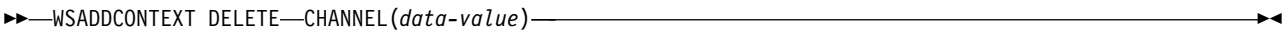
20 This condition occurs when the length of the stored data is greater than the value specified by the EPRLENGTH option. This condition only applies to the EPRINTO option and cannot occur with the EPRSET option.

Default action: ends the task abnormally.

WSACONTEXT DELETE

Use the **WSACONTEXT DELETE** command to delete the addressing context.

WSACONTEXT DELETE



Conditions: CHANNELERR, NOTFND, INVREQ

This command is threadsafe.

Description

The **WSACONTEXT DELETE** command deletes the addressing contexts for the request and response SOAP messages from the specified channel. You can use this command only in a web service requester.

Options

CHANNEL(*data-value*)

Specifies the name of the channel holding the addressing context. The name of the channel can be up to 16 characters in length. If the channel name is less than 16 characters, you must pad the data value with trailing blanks. You can specify the channel name DFHTRANSACTION to use the transaction channel.

Conditions

122 CHANNELERR
RESP2 values:

1 The name specified for the CHANNEL option contains an incorrect character or combination of characters.

2 The specified channel was not located.

16 INVREQ
RESP2 values:

5 This command is not allowed in a web service provider.

13 NOTFND
RESP2 values:

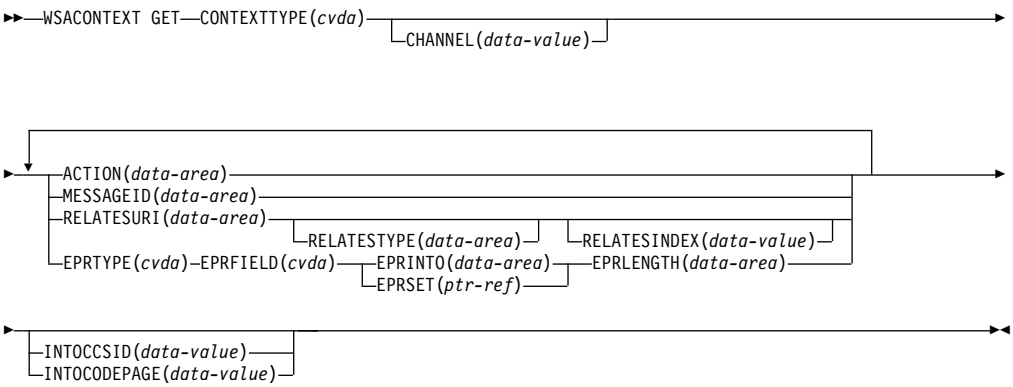
3 The addressing context was not located on the specified channel.

WSACONTEXT GET

Use the **WSACONTEXT GET** command in a service provider to get the message addressing properties (MAPs) sent by the service requester. Use the **WSACONTEXT GET** command in a service requester to get the MAPs of the reply message.

This command cannot be used by Axis2 applications hosted in an Axis2 pipeline in CICS.

WSACONTEXT GET



Conditions: CCSIDERR, CHANNELERR, CODEPAGEERR, INVREQ, LENGERR, NOTFND

This command is threadsafe.

Description

The **WSACONTEXT GET** command is used in a service provider to get the MAPs of the requester from the addressing context, or in a service requester to get the MAPs of the provider from the response message. The **WSACONTEXT GET** command can be called repeatedly to return different MAPs.

Options

ACTION(*data-area*)
Specifies an output area to contain the Action MAP of the request or response SOAP message. The data area must be 255 characters in length. If the Action MAP is fewer than 255 characters, CICS pads the data area with trailing blanks.

CHANNEL(*data-value*)
Specifies the name of the channel that holds the addressing context. The name of the channel can be up to 16 characters in length. If the channel name is fewer than 16 characters, you must pad the data value with trailing blanks. If you do not specify this option, the current channel is implied.

Acceptable characters for the channel name are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blank characters are not permitted. The accepted set of characters for channel names includes some characters that do not have the same representation in all EBCDIC code pages. It is therefore recommended that, if channels are to be shipped between regions, the characters used in naming them should be restricted to A-Z a-z 0-9 & : = , ; < > . - and _.

You can specify the channel name DFHTRANSACTION to use the transaction channel.

This option is required when the **WSACONTEXT GET** command is used by a web service requester application and is the channel name used by subsequent **INVOKE SERVICE** commands.

CONTEXTTYPE(*cvda*)
Specifies which type of addressing context to fetch the MAPs from. CVDA values are as follows:

REQCONTEXT
Addressing context containing the request. Either a web service requester, or a web service provider application can access the MAPs in this addressing context.

RESPCONTEXT
Addressing context containing the response. Only a web service requester application can access the MAPs in this addressing context.

EPRFIELD(*cvda*)
Specifies the part of the endpoint reference that is to be returned in the EPRINTO data area. CVDA values are as follows:

ADDRESS
Return the Address field of the endpoint reference.

ALL Return the complete endpoint reference in XML.

METADATA
Return the Metadata section of the endpoint reference in XML.

REFPARMS
Return the ReferenceParameters section of the endpoint reference in XML.

EPRINTO(*data-area*)
An output data area used to contain the complete or partial endpoint reference. The EPRINTO and EPRSET options are mutually exclusive.

If the EPRINTO option contains an address, any special characters in the address are automatically escaped or unescaped.

The following characters in an address are replaced with an escape sequence: ", >, <, ', &

EPRLENGTH(*data-area*)
Specifies the length, as a halfword binary value, of the endpoint reference.

If you specify the EPRINTO option, you must specify a value for EPRLENGTH unless the length can be generated by the compiler from the EPRINTO option.

The EPRLENGTH option defines the maximum length of data that the program accepts. If the value specified is less than zero, zero is assumed. If the length of the data exceeds the value specified, the data is truncated to that value and the LENGERR condition occurs.

EPRSET(*ptr-ref*)
Specifies the pointer reference that is set to the address of the output data area used to contain the complete or partial endpoint reference. The pointer reference, unless changed by other commands or statements, is valid until the next **WSACONTEXT GET** command or the end of the task. The EPRINTO and EPRSET options are mutually exclusive.

EPRTYPE(*cvda*)
Specifies the type of endpoint reference that is being requested. CVDA values are as follows:

TOEPR
The destination endpoint reference to which a SOAP message is sent.

REPLYTOEPR
An endpoint reference to which a SOAP response message is returned.

FAULTTOEPR
An endpoint reference to which a SOAP fault message is returned.

FROMEPR
An endpoint reference that represents the sender of the SOAP message.

INTOCCSID(*data-value*)
Specifies the Coded Character Set Identifier (CCSID) into which the character data in the addressing context is to be converted, as a fullword binary number. If you prefer to specify an IANA name for the code page, or if you prefer to specify the CCSID as alphanumeric characters, use the INTOCODEPAGE option instead.

For CICS Transaction Server for z/OS applications, the CCSID is typically an EBCDIC CCSID.

For an explanation of CCSIDs, and a list of the CCSIDs supported by CICS, see CICS-supported conversions.

INTOCODEPAGE(*data-value*)
Specifies an IANA-registered alphanumeric charset name or a Coded Character Set Identifier (CCSID) for the code page into which the character data in the addressing context is to be converted, using up to 40 alphanumeric characters, including appropriate punctuation. Use this option instead of the CCSID option if you prefer to use an IANA-registered charset name, as specified in the Content-Type header for an HTTP request. CICS converts the IANA name into a CCSID, and the subsequent data conversion process is identical. Also

use this option if you prefer to specify the CCSID in alphanumeric characters, rather than as a fullword binary number.

Where an IANA name exists for a code page and CICS supports its use, the name is listed with the CCSID in CICS-supported conversions.

MESSAGEID(*data-area*)
Specifies an output area to contain the MessageID MAP of the request or response SOAP message. The data area must be 255 characters in length. If the MessageID MAP is less than 255 characters, CICS pads the data area with trailing blanks.

RELATESINDEX(*data-value*)
Specifies a numeric value that indicates which RelatesTo MAP to return. If this value is not specified, the first RelatesTo MAP is returned. The index starts at 1. If the value of the index is greater than the number of RelatesTo MAPs, spaces are returned in the RELATESTYPE and RELATESURI data areas.

RELATESTYPE(*data-area*)
Specifies an output area that contains a URI denoting the relationship type between this message and another message. The data area must be 255 characters in length. If the URI is less than 255 characters, CICS pads the data area with trailing blanks.

RELATESURI(*data-area*)
Specifies an output area that contains the RelatesTo MAP relationship URI between this message and another. The data area must be 255 characters in length. If the MessageID MAP is less than 255 characters, CICS pads the data area with trailing blanks.

Conditions

123 CCSIDERR
RESP2 values:

1 The CCSID specified on the INTOCCSID option is outside the range of valid CCSID values.

2 The CCSID specified on the INTOCCSID option and the CCSID of the addressing context are an unsupported combination.

4 One or more characters could not be converted. Each unconverted character has been replaced by a blank in the converted data.

5 There was an internal error in the code page conversion of the addressing context data.

122 CHANNELERR
RESP2 values:

1 The name specified for the CHANNEL option contains an incorrect character or combination of characters.

2 The channel specified was not located.

125 CODEPAGEERR
RESP2 values:

1 The code page specified on the INTOCODEPAGE option is not supported.

2 The code page specified on the INTOCODEPAGE option and the CCSID of the addressing context are an unsupported combination.

4 One or more characters could not be converted. Each unconverted character has been replaced by a blank in the converted data.

5 There was an internal error in the code page conversion of a container.

16 INVREQ
RESP2 values:

4 The CHANNEL option was not specified. There is no current channel because the program that issued the command was not passed the name of a channel.

11 The RELATESINDEX option is not valid.

12 The RELATESINDEX option is greater than the number of RelatesTo MAPs.

22 LENGERR
RESP2 values:

20 This condition occurs when the length of the stored data is greater than the value specified by the EPRLENGTH option. This condition only applies to the EPRINTO option and cannot occur with the EPRSET option.

Default action: ends the task abnormally.

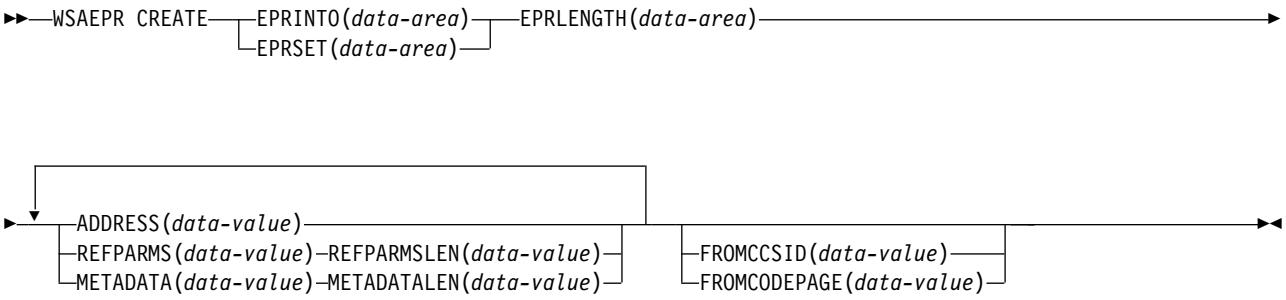
13 NOTFND
RESP2 values:

3 The addressing context was not located.

WSAEPR CREATE

Use the **WSAEPR CREATE** command to create an endpoint reference (EPR) to represent a web service or web service resource.

WSAEPR CREATE



Conditions: CCSIDERR, CODEPAGEERR, INVREQ, LENGERR

This command is threadsafe.

Description

The **WSAEPR CREATE** command creates an endpoint reference, which can represent a web service or web service resource. You can send this EPR to a client so that the addressing context is used for requests to the service.

Options

ADDRESS(*data-value*)
Specifies a URI that forms the address of the endpoint reference. The value of this option must be 255 characters in length. If the URI is less than 255 characters, you must pad the data value with trailing blanks.

The default address of `http://www.w3.org/2005/08/addressing/anonymous` returns information to the caller of the command. Use an address of `http://www.w3.org/2005/08/addressing/none` if no request or response is required. If the To EPR contains a URI, the SOAP message is sent to this URI. If the ReplyTo or FaultTo EPRs contain a URI, response messages are sent to the web service using that URI and not back to the sender of the request message.

EPRINTO(*data-area*)
Specifies the data area used to contain the generated endpoint reference. The EPRINTO and EPRSET options are mutually exclusive.

If the EPRINTO option contains an address, any special characters in the address are automatically escaped or unescaped.

The following characters in an address are replaced with an escape sequence: `", >, <, ', &`

EPRLENGTH(*data-area*)
Specifies the length, as a halfword binary value, of the endpoint reference.

If you specify the EPRINTO option, you must specify a value for EPRLENGTH unless the length can be generated by the compiler from the EPRINTO option.

The EPRLENGTH option defines the maximum length of data that the program accepts. If the value specified is less than zero, zero is assumed. If the length of the data exceeds the value specified, the data is truncated to that value and the LENGERR condition occurs.

EPRSET(*ptr-ref*)
Specifies the pointer reference that is set to the address of the output data area used to contain the complete or partial endpoint reference. The pointer reference, unless changed by other commands or statements, is valid until the next **WSACONTEXT GET** command or the end of the task. The EPRINTO and EPRSET options are mutually exclusive.

FROMCCSID(*data-value*)
Specifies the current Coded Character Set Identifier (CCSID) of the character data to be read or written as a fullword binary number. If you prefer to specify an IANA name for the code page, or if you prefer to specify the CCSID as alphanumeric characters, use the FROMCODEPAGE option instead.

For CICS Transaction Server for z/OS applications, the CCSID is typically an EBCDIC CCSID.

The default CCSID of the region is specified on the LOCALCCSID system initialization option.

For an explanation of CCSIDs, and a list of the CCSIDs supported by CICS, see CICS-supported conversions.

FROMCODEPAGE(*data-value*)
Specifies an IANA-registered alphanumeric charset name or a Coded Character Set Identifier (CCSID) for the current code page of the character data to be

read or written using up to 40 alphanumeric characters, including appropriate punctuation. Use this option instead of the CCSID option if you prefer to use an IANA-registered charset name, as specified in the Content-Type header for an HTTP request. CICS converts the IANA name into a CCSID, and the subsequent data conversion process is identical. Also use this option if you prefer to specify the CCSID in alphanumeric characters, rather than as a fullword binary number.

Where an IANA name exists for a code page and CICS supports its use, the name is listed with the CCSID in CICS-supported conversions.

METADATA(*data-value*)
Specifies metadata that describes the behavior, policies, and capabilities of the endpoint targeted by the endpoint reference. The metadata must be described in XML.

METADATALEN(*data-value*)
Specifies the length, as fullword binary, of the METADATA option.

REFPARMS(*data-value*)
Specifies application reference options that form part of the endpoint reference. These options are described in XML.

REFPARMSLEN(*data-value*)
Specifies the length, as fullword binary, of the reference options.

Conditions

123 CCSIDERR
RESP2 values:

1 The CCSID specified on the FROMCCSID option is outside the range of valid CCSID values.

2 The CCSID specified on the FROMCCSID option and the CCSID of the container are an unsupported combination.

5 There was an internal error in the code page conversion of a container.

6 Either the text encoding is not compatible with the specified CCSID on the FROMCCSID option, or one or more characters are truncated.

125 CODEPAGEERR
RESP2 values:

1 The code page specified on the FROMCODEPAGE option is not supported.

2 The code page specified on the FROMCODEPAGE option and the CCSID of the container are an unsupported combination.

5 There was an internal error in the code page conversion of a container.

6 Either the text encoding is not compatible with the specified CODEPAGE on the FROMCODEPAGE option, or one or more characters are truncated.

16 INVREQ
The INVREQ RESP2 values and the corresponding messages are shown below. For this command, if the EIBRESP2 value is > 100, the fullword EIBRESP2 field is regarded as a structure containing two halfwords. The low-order halfword always contains the error number. The high-order halfword contains the offset into the XML data where the parsing error occurred.

RESP2 values:

8 One of the input parameters contains an incorrect value. The ADDRESS option might not contain a valid URI.

13 One of the input parameters contains an incorrect value. The METADATA option might not contain valid XML.

14 One of the input parameters contains an incorrect value. The REFPARMS option might not contain valid XML.

22 LENGERR

RESP2 values:

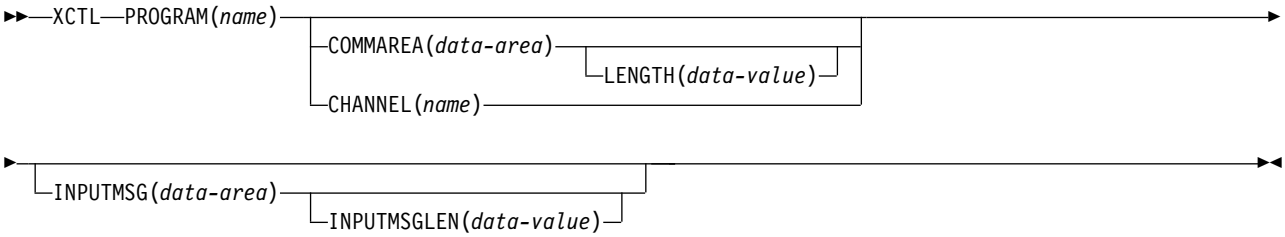
20 This condition occurs when the length of the stored data is greater than the value specified by the EPRLENGTH option. This condition only applies to the EPRINTO option and cannot occur with the EPRSET option.

Default action: ends the task abnormally.

XCTL

Transfer program control.

XCTL



Conditions: CHANNELERR, INVREQ, LENGERR, NOTAUTH, PGMIDERR

This command is threadsafe.

Description

XCTL transfers control from one application program to another at the same logical level. The program from which control is transferred is released. If the program to which control is transferred is not already in main storage, it is loaded.

This command operates in the current application context. If the command is issued by a program that is running under a task for an application deployed on a platform, CICS searches for the named program in the private program directory for the application. This command cannot be used to change application context.

Options

CHANNEL(*name*)

Specifies the name (1–16 characters) of a channel that is to be made available to the invoked program. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = ¬ , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with

trailing blanks up to 16 characters. If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

Channel names are always in EBCDIC. The set of allowed characters for channel names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be shipped between regions, it is advisable to restrict the characters used to name them to A-Z a-z 0-9 & : = , ; < > . - and _.

You can specify the channel name DFHTRANSACTION to use a transaction channel. A transaction channel does not go out of scope when the link level changes: it is always accessible in the task. For more information, see Channels and containers.

The program that issues the XCTL command can do one of the following:
- Create the channel before issuing the XCTL command, by using one or more **PUT CONTAINER CHANNEL** or **PUT64 CONTAINER** commands.
- Specify its current channel, by name.
- Name a channel that does not currently exist. A new empty channel is created.

COMMAREA(*data-area***)**
Specifies a communication area to be made available to the invoked program. In this option the contents of the data-area are passed. In COBOL, you must give this data area the name DFHCOMMAREA in the receiving program. See Passing data to other programs.

INPUTMSG(*data-area***)**
Specifies data to be passed to the invoked program when it first issues a RECEIVE command. If the invoked program passes control to another program by using a **LINK** command, a linked chain is created, as described under the INPUTMSG option of the **LINK** command. The INPUTMSG data remains available until a **RECEIVE** command is issued or until control returns to CICS.

INPUTMSGLEN(*data-value***)**
Specifies a halfword binary value that specifies the length of the data passed by INPUTMSG.

LENGTH(*data-value***)**
Specifies the length (halfword binary data value) in bytes of the communication area. For a description of a suitable upper limit, see "LENGTH options in CICS commands" on page 10.

PROGRAM(*name***)**
Specifies the identifier (1–8 alphanumeric characters) of the program to which control is to be passed unconditionally. The specified name must have been defined as a program to CICS, though if AUTOINSTALL is active a definition is autoinstalled.

Note the use of quotes:
PROGX is in quotes because it is the program name.

```
EXEC CICS LINK PROGRAM('PROGX')
```

DAREA is not in quotes because it is the name of a data area that contains the

```
EXEC CICS LINK PROGRAM(DAREA)
```

actual program name. It a data area is used to contain the program name, this data area must be defined as an 8 byte field in working storage.

Conditions

122 CHANNELERR

RESP2 values:

1 The name specified on the CHANNEL option contains an illegal character or combination of characters.

16 INVREQ

RESP2 values:

8 An XCTL command with the INPUTMSG option is issued for a program that is not associated with a terminal, or that is associated with an APPC logical unit, or an IRC session.

29 EXEC XCTL is not allowed in a GLUE or TRUE.

30 The program manager domain has not yet been initialized. This is probably because an XCTL request was made in a first stage PLT.

31 An XCTL command is issued from a program that is running with an application context, to another program that is an application entry point.

33 An XCTL command is issued from a public program to another program that is an application entry point.

34 An XCTL command is issued from a program that is running with an application context, to a public program.

200 An XCTL command with the INPUTMSG option is issued in a program invoked by DPL.

Default action: terminate the task abnormally.

22 LENGERR

RESP2 values:

11 LENGTH is less than 0 or greater than 32763.

26 The COMMAREA address passed was zero, but LENGTH was non-zero.

27 INPUTMSGLEN was less than 0 or greater than 32767.

28 LENGTH or INPUTMSGLEN is greater than the length of the data area specified in the COMMAREA or INPUTMSG options, and while that data was being copied, a destructive overlap occurred because of the incorrect length.

Default action: terminate the task abnormally.

70 NOTAUTH

RESP2 values:

101 A resource security check has failed on PROGRAM(name).

Default action: terminate the task abnormally.

27 PGMIDERR

RESP2 values:

1 A program has no installed resource definition and either program autoinstall was switched off, or the program autoinstall control program indicated that the program should not be autoinstalled.

2 The program is disabled.

3 A program could not be loaded for one of the following reasons:

- This was the first load of the program and the program load failed, usually because the load module could not be found.
- This was a subsequent load of the program, but the first load failed.

To reset the load status, the load module must be in the DFHRPL or dynamic LIBRARY concatenation, and a SET PROGRAM NEWCOPY will be required

9 The installed program definition is for a remote program.

21 The program autoinstall control program failed, either because the program autoinstall control program is incorrect or incorrectly defined, or as a result of an abend in the program autoinstall control program. Program autoinstall is disabled and message DFHPG0202 or DFHPG0203 written to the CSPL.

22 The model returned by the program autoinstall control program was not defined to CICS, or was not enabled.

23 The program autoinstall control program returned invalid data.

24 Define for the program failed because autoinstall returned an invalid program name or definition.

Default action: terminate the task abnormally.

Examples

The following example shows how to request a transfer of control to an application program called PROG2:

```
EXEC CICS XCTL PROGRAM('PROG2')
```

Chapter 3. Threadsafe commands

The commands that are threadsafe, or threadsafe in certain conditions, are listed.

Not all **EXEC CICS** commands are threadsafe, and issuing any of the non-threadsafe commands causes CICS to use the QR TCB to ensure serialization. See Threadsafe programs for information about writing threadsafe application programs.

In the following list of threadsafe commands, an asterisk (*) indicates commands that are threadsafe only in certain conditions:

- These program link, file control, temporary storage, and transient data commands are threadsafe in the following circumstances:
  - The program, file, or queue to which the command refers is defined as local.
  - The program, file, or queue to which the command refers is defined as remote, and the resource is accessed by distributed program link or function shipping to a remote CICS region over an IPIC connection.
  - For file control commands, as an additional condition, the file to which the command refers is VSAM in RLS mode.
- These commands are not threadsafe in the following circumstances:
  - The resource is accessed by distributed program link or function shipping to a remote CICS region over another type of connection.
  - For file control commands, the file to which the command refers is a shared data table, coupling facility data table, or BDAM file.
- The **WRITE OPERATOR** command is threadsafe, but cannot be used from a key 9 TCB if the REPLY option is specified. In this case CICS will switch to the QR TCB to process the command.

Invoking DL/I by using the applicable language interface, for example the COBOL statement **CALL CBLTDLI**, is threadsafe when used with IMS Version 12 or later.

Threadsafe command list

- **ABEND**
- **ADDRESS**
- **ASKTIME**
- **ASSIGN**
- **BIF DEEDIT**
- **BIF DIGEST**
- **CHANGE PASSWORD**
- **CHANGE PHRASE**
- **CHANGE TASK**
- **CONVERTTIME**
- **DEFINE COUNTER** and **DEFINE DCOUNTER**
- **DELETE** *
- **DELETE CHANNEL**
- **DELETE CONTAINER (CHANNEL)**
- **DELETE COUNTER** and **DELETE DCOUNTER**
- **DELETEQ TD***

© Copyright IBM Corp. 1989, 2017

- **DELETEQ TS***
- **DEQ** (This command is threadsafe if it is defined as LOCAL. It is non-threadsafe if it is defined as GLOBAL.)
- **DOCUMENT CREATE**
- **DOCUMENT DELETE**
- **DOCUMENT INSERT**
- **DOCUMENT RETRIEVE**
- **DOCUMENT SET**
- **ENDBR** *
- **ENQ** (This command is threadsafe if it is defined as LOCAL. It is non-threadsafe if it is defined as GLOBAL.)
- **ENTER TRACENUM**
- **EXEC DLI**
- **EXTRACT CERTIFICATE**
- **EXTRACT TCPIP**
- **EXTRACT WEB**
- **FETCH ANY**
- **FETCH CHILD**
- **FORMATTIME**
- **FREEMAIN**
- **FREEMAIN64**
- **GET CONTAINER (CHANNEL)**
- **GET COUNTER** and **GET DCOUNTER**
- **GETMAIN**
- **GETMAIN64**
- **GET64 CONTAINER**
- **HANDLE ABEND**
- **HANDLE AID**
- **HANDLE CONDITION**
- **IGNORE CONDITION**
- **INVOKE APPLICATION**
- **INVOKE SERVICE**
- **INVOKE WEBSERVICE**
- **LINK** *
- **LOAD**
- **MONITOR**
- **MOVE CONTAINER (CHANNEL)**
- **POP HANDLE**
- **PUSH HANDLE**
- **PUT CONTAINER (CHANNEL)**
- **PUT64 CONTAINER**
- **QUERY CHANNEL**
- **QUERY COUNTER** and **QUERY DCOUNTER**
- **QUERY SECURITY**
- **READ** *

- **READNEXT** *
- **READPREV** *
- **READQ TD***
- **READQ TS***
- **RELEASE**
- **REQUEST ENCYRPTPTKT**
- **REQUEST PASSTICKET**
- **RESETBR** *
- **RETURN**
- **REWIND COUNTER** and **REWIND DCOUNTER**
- **REWRITE** *
- **RUN TRANSID**
- **SIGNAL EVENT**
- **SIGNOFF**
- **SIGNON**
- **SIGNON TOKEN**
- **SOAPFAULT ADD**
- **SOAPFAULT CREATE**
- **SOAPFAULT DELETE**
- **STARTBR** *
- **SUSPEND**
- **SYNCPOINT** (The Recovery Manager processes this command on an open TCB wherever possible to minimize TCB switching.)
- **SYNCPOINT ROLLBACK** (The Recovery Manager processes this command on an open TCB wherever possible to minimize TCB switching.)
- **TRANSFORM DATATOJSON**
- **TRANSFORM DATATOXML**
- **TRANSFORM JSONTODATA**
- **TRANSFORM XMLTODATA**
- **UNLOCK** *
- **UPDATE COUNTER** and **UPDATE DCOUNTER**
- **VERIFY PASSWORD**
- **VERIFY PHRASE**
- **VERIFY TOKEN**
- **WAIT EXTERNAL**
- **WAIT JOURNALNAME**
- **WAIT JOURNALNUM**
- **WEB CLOSE**
- **WEB CONVERSE**
- **WEB ENDBROWSE FORMFIELD**
- **WEB ENDBROWSE HTTPHEADER**
- **WEB ENDBROWSE QUERYPARM**
- **WEB EXTRACT**
- **WEB OPEN**
- **WEB PARSE URL**

- **WEB READ FORMFIELD**
- **WEB READ HTTPHEADER**
- **WEB READNEXT FORMFIELD**
- **WEB READNEXT HTTPHEADER**
- **WEB READ QUERYPARM**
- **WEB READNEXT QUERYPARM**
- **WEB RECEIVE**
- **WEB RETRIEVE**
- **WEB SEND**
- **WEB STARTBROWSE FORMFIELD**
- **WEB STARTBROWSE HTTPHEADER**
- **WEB STARTBROWSE QUERYPARM**
- **WEB WRITE HTTPHEADER**
- **WRITE** *
- **WRITE JOURNALNAME**
- **WRITE JOURNALNUM**
- **WRITE OPERATOR***
- **WRITEQ TD***
- **WRITEQ TS***
- **WSACONTEXT BUILD**
- **WSACONTEXT DELETE**
- **WSACONTEXT GET**
- **WSAEPR CREATE**
- **XCTL**

Chapter 4. EXEC interface block fields

An application program can read all the fields in the EXEC interface block (EIB) of the associated task by name. An application must not change the contents of any of the fields, other than through an **EXEC CICS** command.

For each EIB field, the contents and format (for each of the application programming languages COBOL, C, PL/I, and Assembler) are given. Fields explained are EIBAID, EIBATT, EIBCALEN, EIBCOMPL, EIBCONF, EIBCPOSN, EIBDATE, EIBDS, EIBEOC, EIBERR, EIBERRCD, EIBFMH, EIBFN, EIBFREE, EIBNODAT, EIBRCODE, EIBRECV, EIBREQID, EIBRESP, EIBRESP2, EIBRLDBK, EIBRSRCE, EIBSIG, EIBSYNC, EIBSYNRB, EIBTASKN, EIBTIME, EIBTRMID, and EIBTRNID.

All fields contain binary zeros in the absence of meaningful information. Fields are listed in alphabetic order.

EIBAID
Contains the attention identifier (AID) associated with the last terminal control or basic mapping support (BMS) input operation from a display device such as the 3270.

```
COBOL:     PIC X(1).
C:         unsigned char eibaid;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBATT
Indicates that the RU contains attach header data (X'FF').

```
COBOL:     PIC X(1).
C:         unsigned char eibatt;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBCALEN
Contains the length of the communication area that has been passed to the application program from the last program, using the COMMAREA and LENGTH options. If no communication area is passed, this field contains binary zeros.

```
COBOL:     PIC S9(4) COMP.
C:         short int eibcalen;
PL/I:      FIXED BIN(15)
Assembler: H
```

EIBCOMPL
Indicates, on a terminal control RECEIVE command, whether the data is complete (X'FF'). If the NOTRUNCATE option has been used on the RECEIVE command, CICS retains data in excess of the amount requested via the

© Copyright IBM Corp. 1989, 2017

LENGTH or MAXLENGTH option. EIBRECV is set indicating that further RECEIVE commands are required. EIBCOMPL is not set until the last of the data has been retrieved.

EIBCOMPL is always set when a RECEIVE command without the NOTRUNCATE option is executed.

```
COBOL:     PIC X(1).
C:         unsigned char eibcompl;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBCONF

Indicates that a CONFIRM request has been received (X'FF') for an APPC conversation.

```
COBOL:     PIC X(1).
C:         unsigned char eibconf;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBCPOSN

Contains the cursor address (position) associated with the last terminal control or basic mapping support (BMS) input operation from a display device such as the 3270.

```
COBOL:     PIC S9(4) COMP.
C:         short int eibcposn;
PL/I:      FIXED BIN(15)
Assembler: H
```

EIBDATE

Contains the date the task is started; this field is updated by the ASKTIME command. The date is in packed decimal form (0CYYDDD+) where C shows the century with values 0 for the 1900s and 1 for the 2000s. For example, the date 31 December 1999 has the EIBDATE value of 0099365 and the date 1 January 2000 has an EIBDATE value of 0100001.

At midnight, if EIBTIME has the value of 0240000+, the value of EIBDATE is the day that has ended. If EIBTIME has the value of 0000000+, the value of EIBDATE is the day that is just beginning.

```
COBOL:     PIC S9(7) COMP-3.
C:         char eibdate [4];
PL/I:      FIXED DEC(7,0)
Assembler: PL4
```

EIBDS

Contains the symbolic identifier of the last data set referred to in a file control request.

```
COBOL:     PIC X(8).
C:         char eibds [8];
PL/I:      CHAR(8)
Assembler: CL8
```

EIBEOC
Indicates that an end-of-chain indicator appears in the RU just received (X'FF').

```
COBOL:     PIC X(1).
C:         unsigned char eibeoc;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBERR
Indicates that an error has been received (X'FF') on an APPC conversation.

```
COBOL:     PIC X(1).
C:         unsigned char eiberr;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBERRCD
When EIBERR is set, contains the error code that has been received. The following values can be returned in the first two bytes of EIBERRCD:

X'0889'
Conversation error detected.

X'0824'
SYNCPOINT ROLLBACK requested.

```
COBOL:     PIC X(4).
C:         char eiberrcd [4];
PL/I:      CHAR(4)
Assembler: CL4
```

See CICS mapping to the APPC architecture for information about other EIBERRCD values that can occur.

EIBFMH
Indicates that the user data received or retrieved contains an FMH (X'FF').

```
COBOL:     PIC X(1).
C:         unsigned char eibfmh;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBFN
Contains a code that identifies the last CICS command issued by the task.

```
COBOL:     PIC X(2).
C:         char eibfn [2];
PL/I:      CHAR(2)
Assembler: CL2
```

Code	Command
0202	ADDRESS
0204	HANDLE CONDITION
0206	HANDLE AID
0208	ASSIGN
020A	IGNORE CONDITION
020C	PUSH HANDLE
020E	POP HANDLE
0210	ADDRESS SET
0402	RECEIVE
0404	SEND
0406	CONVERSE
0408	ISSUE EODS
040A	ISSUE COPY
040C	WAIT TERMINAL
040E	ISSUE LOAD
0410	WAIT SIGNAL
0412	ISSUE RESET
0414	ISSUE DISCONNECT
0416	ISSUE ENDOUTPUT
0418	ISSUE ERASEAUP
041A	ISSUE ENDFILE
041C	ISSUE PRINT
041E	ISSUE SIGNAL
0420	ALLOCATE
0422	FREE
0424	POINT
0426	BUILD ATTACH
0428	EXTRACT ATTACH
042A	EXTRACT TCT
042C	WAIT CONVID
042E	EXTRACT PROCESS
0430	ISSUE ABEND
0432	CONNECT PROCESS
0434	ISSUE CONFIRMATION
0436	ISSUE ERROR
0438	ISSUE PREPARE
043A	ISSUE PASS
043C	EXTRACT LOGONMSG
043E	EXTRACT ATTRIBUTES
0602	READ
0604	WRITE
0606	REWRITE
0608	DELETE
060A	UNLOCK
060C	STARTBR
060E	READNEXT
0610	READPREV
0612	ENDBR

0614 RESETBR

0802 WRITEQ TD
0804 READQ TD
0806 DELETEQ TD

0A02 WRITEQ TS
0A04 READQ TS
0A06 DELETEQ TS

0C02 GETMAIN
0C04 FREEMAIN
0C12 GETMAIN64
0C14 FREEMAIN64

0E02 LINK
0E04 XCTL
0E06 LOAD
0E08 RETURN
0E0A RELEASE
0E0C ABEND
0E0E HANDLE ABEND
0E10 INVOKE APPLICATION

1002 ASKTIME
1004 DELAY
1006 POST
1008 START
1008 START ATTACH
1008 START BREXIT
100A RETRIEVE
100C CANCEL

1202 WAIT EVENT
1204 ENQ
1206 DEQ
1208 SUSPEND

1402 WRITE JOURNALNUM
1404 WAIT JOURNALNUM
1406 WRITE JOURNALNAME
1408 WAIT JOURNALNAME

1602 SYNCPOINT

1802 RECEIVE MAP
1804 SEND MAP
1806 SEND TEXT
1808 SEND PAGE
180A PURGE MESSAGE
180C ROUTE
180E RECEIVE PARTN
1810 SEND PARTNSET
1812 SEND CONTROL

1A02 TRACE

1A04 ENTER TRACEID

1C02 DUMP

1E02 ISSUE ADD
1E04 ISSUE ERASE
1E06 ISSUE REPLACE
1E08 ISSUE ABORT
1E0A ISSUE QUERY
1E0C ISSUE END
1E0E ISSUE RECEIVE
1E10 ISSUE NOTE
1E12 ISSUE WAIT
1E14 ISSUE SEND

2002 BIF DEEDIT

2004 DEFINE COUNTER
2006 GET COUNTER
2008 UPDATE COUNTER
200A DELETE COUNTER
200C REWIND COUNTER
200E QUERY COUNTER
2014 DEFINE DCOUNTER
2016 GET DCOUNTER
2018 UPDATE DCOUNTER
201A DELETE DCOUNTER
201C REWIND DCOUNTER
201E QUERY DCOUNTER

2020 BIF DIGEST
2602 TRANSFORM DATATOXML
2604 TRANSFORM XMLTODATA
| 2606 TRANSFORM DATATOJSON
| 2608 TRANSFORM JSONTODATA

2802 SIGNAL EVENT

3402 DEFINE ACTIVITY
3404 DEFINE PROCESS
3406 RUN ACTIVITY
3408 RUN ACQPROCESS
340E ACQUIRE PROCESS
3410 ACQUIRE ACTIVITYID
3412 DELETE CONTAINER
3414 GET CONTAINER
3416 PUT CONTAINER
3418 RESET ACTIVITY
341A CHECK ACTIVITY
341C CANCEL ACTIVITY
341E CANCEL ACQPROCESS
3420 SUSPEND ACTIVITY
3422 SUSPEND ACQPROCESS
3424 RESUME ACTIVITY
3426 RESUME ACQPROCESS
3428 DELETE ACTIVITY

342A LINK ACQPROCESS
342C LINK ACTIVITY
342E CANCEL ACQACTIVITY
3430 RUN ACQACTIVITY
3432 LINK ACQACTIVITY
3434 SUSPEND ACQACTIVITY
3436 RESUME ACQACTIVITY
3438 CHECK ACQPROCESS
343A CHECK ACQACTIVITY
343C RESET ACQPROCESS
343E RUN TRANSID
3440 MOVE CONTAINER
3442 FETCH CHILD
3444 FETCH ANY
3446 FREE CHILD
3454 GET64 CONTAINER
3456 PUT64 CONTAINER
3458 DELETE CHANNEL
345A QUERY CHANNEL

3602 DEFINE INPUT EVENT
3602 DEFINE COMPOSITE EVENT
3604 DELETE EVENT
3608 ADD SUBEVENT
360A REMOVE SUBEVENT
360E TEST EVENT
3610 RETRIEVE REATTACH EVENT
3612 RETRIEVE SUBEVENT
3614 DEFINE TIMER
3616 DELETE TIMER
3618 CHECK TIMER
361A FORCE TIMER

3802 WEB RECEIVE
3804 WEB SEND
3806 WEB READ
3808 WEB STARTBROWSE
380A WEB READNEXT
380C WEB ENDBROWSE
380E WEB WRITE HTTPHEADER
3810 WEB EXTRACT
3814 WEB RETRIEVE
3816 WEB PARSE URL
3818 WEB OPEN
381A WEB CLOSE
381C WEB CONVERSE

3C02 DOCUMENT CREATE
3C04 DOCUMENT INSERT
3C06 DOCUMENT RETRIEVE
3C08 DOCUMENT SET
3C10 DOCUMENT DELETE

3E0E EXTRACT TCPIP
3E10 EXTRACT CERTIFICATE

4802 ENTER TRACENUM
4804 MONITOR

4A02 ASKTIME ABSTIME
4A04 FORMATTIME
4A06 CONVERTTIME

5602 SPOOLOPEN
5604 SPOOLREAD
5606 SPOOLWRITE
5610 SPOOLCLOSE

5E06 CHANGE TASK
5E22 WAIT EXTERNAL
5E32 WAITCICS

6A02 QUERY SECURITY
6C02 WRITE OPERATOR
6C12 CICSMESSAGE

7402 SIGNON
7404 SIGNOFF
7406 VERIFY PASSWORD
7408 CHANGE PASSWORD
740A VERIFY PHRASE
740C CHANGE PHRASE
740E VERIFY TOKEN
7410 REQUEST PASSTICKET

7E02 DUMP TRANSACTION

8C02 WRITE MESSAGE

C002 INVOKE WEBSERVICE
C004 SOAPFAULT CREATE
C006 SOAPFAULT ADD
C008 SOAPFAULT DELETE

C00A WSACONTEXT BUILD
C00C WSACONTEXT GET
C00D WSACONTEXT DELETE
C010 WSAEPR CREATE

EIBFREE
Indicates that the application program cannot continue using the facility. The application program should either free the facility or should terminate so that the facility is freed by CICS (X'FF').

```
COBOL:     PIC X(1).
C:         unsigned char eibfree;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBNODAT
Indicates that no data has been sent by the remote application (X'FF'). A

message has been received from the remote system that conveyed only control information. For example, if the remote application executed a SEND command with the WAIT option, any data would be sent across the link. If the remote application then executed a SEND INVITE command without using the FROM option to transmit data at the same time, it would be necessary to send the INVITE instruction across the link by itself. In this case, the receiving application finds EIBNODAT set. The use of this field is restricted to application programs holding conversations across APPC links only.

```
COBOL:     PIC X(1).
C:         unsigned char eibnodat;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBRCODE

Contains the CICS response code returned after the function requested by the last CICS command to be issued by the task has been completed.

Note: For commands where EIBRESP and EIBRESP2 are used for interrogating the resulting condition of an executed command, byte 3 of EIBRCODE has the same value as EIBRESP. Any further information is in EIBRESP2 rather than EIBRCODE. For a normal response, this field contains hexadecimal zeros (6 X'00').

Almost all of the information in this field can be used within application programs by the **HANDLE CONDITION** command.

```
COBOL:     PIC X(6).
C:         char eibrcode [6];
PL/I:      CHAR(6)
Assembler: CL6
```

The following list contains the values of the bytes together with the names of the conditions associated with the return codes.

See the notes at the end of the list of values for explanations of the numbers following some of the conditions.

EIBFN	EIBRCODE	Condition
02 ..	E0	INVREQ
04 ..	04	EOF
04 ..	10	EODS
04 ..	C1	EOF
04 ..	C2	ENDINPT
04 ..	D0	SYSIDERR (see note 1)
04 ..	D2	SESSIONERR (see note 2)
04 ..	D3	SYSBUSY (see note 3)
04 ..	D4	SESSBUSY
04 ..	D5	NOTALLOC
04 ..	E0	INVREQ (see note 4)
04 ..	E1	LENGERR (see note 5)
04 ..	E3	WRBRK
04 ..	E4	RDATT
04 ..	E5	SIGNAL

EIBFN	EIBRCODE	Condition
04 ..	E6	TERMIDERR
04 ..	E7	NOPASSBKRD
04 ..	E8	NOPASSBKWR
04 ..	EA	IGREQCD
04 ..	EB	CBIDERR
04 ..	EC	PARTNERIDERR
04 ..	ED	NETNAMEIDERR
04 ..	F1	TERMERR
04 ..	.. 20	EOC
04 ..	.. 40	INBFMH
04 ..	.. .. .. F6	NOSTART
04 ..	.. .. .. F7	NONVAL
06 ..	01	FILENOTFOUND
06 ..	02	ILLOGIC (see note 6)
06 ..	03	LOCKED
06 ..	05	RECORDBUSY
06 ..	08	INVREQ
06 ..	0C	NOTOPEN
06 ..	0D	DISABLED
06 ..	0F	ENDFILE
06 ..	80	IOERR (see note 6)
06 ..	81	NOTFND
06 ..	82	DUPREC
06 ..	83	NOSPACE
06 ..	84	DUPKEY
06 ..	85	SUPPRESSED
06 ..	86	LOADING
06 ..	D0	SYSIDERR (see note 1)
06 ..	D1	ISCINVREQ
06 ..	D6	NOTAUTH
06 ..	E1	LENGERR
08 ..	01	QZERO
08 ..	02	QIDERR
08 ..	04	IOERR
08 ..	08	NOTOPEN
08 ..	10	NOSPACE
08 ..	C0	QBUSY
08 ..	D0	SYSIDERR (see note 1)
08 ..	D1	ISCINVREQ
08 ..	D6	NOTAUTH
08 ..	D7	DISABLED
08 ..	E0	INVREQ
08 ..	E1	LENGERR
0A ..	01	ITEMERR
0A ..	02	QIDERR
0A ..	04	IOERR
0A ..	08	NOSPACE
0A ..	20	INVREQ
0A ..	D0	SYSIDERR (see note 1)
0A ..	D1	ISCINVREQ
0A ..	D6	NOTAUTH

EIBFN	EIBRCODE	Condition
0A ..	E1	LENGERR
0C ..	E0	INVREQ
0C ..	E1	LENGERR
0C ..	E2	NOSTG
0E ..	01	PGMIDERR
0E ..	D6	NOTAUTH
0E ..	D9	RESUNAVAIL
0E ..	DA	CHANNELERR
0E ..	E0	INVREQ
0E ..	E1	LENGERR
0E ..	F1	TERMERR
10 ..	01	ENDDATA
10 ..	04	IOERR
10 ..	11	TRANSIDERR
10 ..	12	TERMIDERR
10 ..	20	EXPIRED
10 ..	81	NOTFND
10 ..	D0	SYSIDERR (see note 1)
10 ..	D1	ISCINVREQ
10 ..	D6	NOTAUTH
10 ..	D8	USERIDERR
10 ..	D9	RESUNAVAIL
10 ..	DA	CHANNELERR
10 ..	E1	LENGERR
10 ..	E9	ENVDEFERR
10 ..	FF	INVREQ
12 ..	32	ENQBUSY
12 ..	E0	INVREQ
12 ..	E1	LENGERR
14 ..	01	JIDERR
14 ..	02	INVREQ
14 ..	05	NOTOPEN
14 ..	06	LENGERR
14 ..	07	IOERR
14 ..	09	NOJBUFSP
14 ..	D6	NOTAUTH
16 ..	01	ROLLEDBACK
18 ..	01	INVREQ
18 ..	02	RETPAGE
18 ..	04	MAPFAIL
18 ..	08	INVMPSZ (see note 7)
18 ..	20	INVERRTERM
18 ..	40	RTESOME
18 ..	80	RTEFAIL
18 ..	E1	LENGERR
18 ..	E3	WRBRK
18 ..	E4	RDATT

EIBFN	EIBRCODE	Condition
18 ..	.. 02	PARTNFAIL
18 ..	.. 04	INVPARTN
18 ..	.. 08	INVPARTNSET
18 ..	.. 10	INVLDC
18 ..	.. 20	UNEXPIN
18 ..	.. 40	IGREQCD
18 ..	.. 80	TSIOERR
18 ..	.. .. 01	OVERFLOW
18 ..	.. .. 04	EODS
18 ..	.. .. 08	EOC
18 ..	.. .. 10	IGREQID
1A ..	E0	INVREQ
1A ..	04	DSSTAT
1A ..	08	FUNCERR
1A ..	0C	SELNERR
1A ..	10	UNEXPIN
1A ..	E1	LENGERR
1A ..	.. 11	EODS
1A ..	.. 2B	IGREQCD
1A ..	.. .. 20	EOC
22 ..	80	INVEXITREQ
4A ..	.. .. .. 01	INVREQ
56 ..	.. .. .. 0D	NOTFND
56 ..	.. .. .. 10	INVREQ
56 ..	.. .. .. 13	NOTOPEN
56 ..	.. .. .. 14	ENDFILE
56 ..	.. .. .. 15	ILLOGIC
56 ..	.. .. .. 16	LENGERR
56 ..	.. .. .. 2A	NOSTG
56 ..	.. .. .. 46	NOTAUTH
56 ..	.. .. .. 50	NOSPOOL
56 ..	.. .. .. 55	ALLOCERR
56 ..	.. .. .. 56	STRELERR
56 ..	.. .. .. 57	OPENERR
56 ..	.. .. .. 58	SPOLBUSY
56 ..	.. .. .. 59	SPOLERR
56 ..	.. .. .. 5A	NODEIDERR

Note:

1. When SYSIDERR occurs, further information is provided in bytes 1 and 2 of EIBRCODE, as shown inFigure 5 on page 767.

```
.. 04 00 .. .. ..   request was for a function
                    that is not valid.
.. 04 04 .. .. ..   no session available and
                    NOQUEUE.
.. 04 08 .. .. ..   modename not found (for APPC only).
.. 04 0C .. .. ..   modename not valid (for APPC only).
.. 04 10 .. .. ..   task canceled or timed
                    out during allocation (for APPC only).
.. 04 14 .. .. ..   mode group is out of
                    service (for APPC only).
.. 04 18 .. .. ..   close - DRAIN=ALL (for APPC only).
.. 08 .. .. .. ..   sysid is not available.
.. 08 00 .. .. ..   no session available,
                    all sessions are out
                    of service, or released,
                    or being quiesced.
.. 08 04 .. .. ..   no session available,
                    request to queue rejected
                    by XZIQUE global user
                    exit program.
.. 08 08 .. .. ..   no session available;
                    request rejected by XZIQUE
                    global user exit program.
.. 0C xx .. .. ..   sysid definition error.
.. 0C 00 .. .. ..   name not that of TCTSE.
.. 0C 04 .. .. ..   name not that of remote
                    TCTSE.
.. 0C 08 .. .. ..   mode name not found.

.. 0C 0C .. .. ..   profile not found.
```

Figure 5. Bytes 1 and 2 of EIBRCODE for SYSIDERR

Further information about SYSIDERR can be found in Syncpoint exchanges.

2. When SESSIONERR occurs, further information is provided in bytes 1 and 2 of EIBRCODE, as shown in Figure 6.

```
.. 08 .. .. .. ..   session out of service
.. 0C xx .. .. ..   session definition error
.. 0C 00 .. .. ..   name not found
.. 0C 0C .. .. ..   profile not found.
```

Figure 6. Bytes 1 and 2 of EIBRCODE for SESSIONERR

Further information about SESSIONERR can be found in CICS-to-IMS applications: DTP.

3. If SYSBUSY occurs on an ALLOCATE command that attempts to acquire a session to an APPC terminal or system, byte 3 of the EIBRCODE indicates where the error condition was detected as shown in Figure 7 on page 768.

```
.. .. .. 00 .. ..   the request was for a
                    session to a connected
                    terminal or system.
.. .. .. 01 .. ..   the request was for a
                    session to a remotely
                    connected terminal or
                    system, and the error
                    occurred in the terminal-
                    owning region (TOR) or
                    an intermediate system.
.. .. .. 02 .. ..   the request was for a
                    session to a remotely
                    connected terminal or
                    system, and the error
                    occurred in the
                    application-owning
                    region (AOR).
```

Figure 7. Byte 3 of EIBRCODE for SYSBUSY

Further information about SYSBUSY can be found in CICS-to-IMS applications: DTP.

4. When INVREQ occurs during terminal control operations, further information is provided in bytes 1 or 3 of EIBRCODE as shown in Figure 8.

```
.. 24 .. .. .. ..   ISSUE PREPARE command -
                    STATE error.
.. .. .. 04 .. ..   ALLOCATE command - TCTTE
                    already allocated.
.. .. .. 08 .. ..   FREE command - TCTTE in
                    wrong state.
.. .. .. 0C .. ..   CONNECT PROCESS command -
                    SYNCLVL 2 requested, but
                    cannot be supported on
                    the session in use.
.. .. .. 10 .. ..   EXTRACT ATTACH command -
                    incorrect data.
.. .. .. 14 .. ..   SEND command - CONFIRM
                    option specified, but
                    conversation not SYNCLVL 1.
.. .. .. 18 .. ..   EXTRACT TCT command -
                    incorrect netname.
.. .. .. 1C .. ..   an incorrect command has
                    been issued for the terminal
                    or logical unit in use.
.. .. .. 20 .. ..   an incorrect command has
                    been issued for the LUTYPE6.2
                    conversation type in use.
.. .. .. 28 .. ..   GETMAIN failure on ISSUE
                    PASS command.
.. .. .. 2C .. ..   Command invalid in DPL
                    environment.
```

Figure 8. Bytes 1 or 3 of EIBRCODE for INVREQ

5. When LENGERR occurs during terminal control operations, further information is provided in byte 1 of EIBRCODE, as shown in Figure 9 on page 769.

```
.. 00 .. .. .. ..   input data is overlong and
                    has been truncated.
.. 04 .. .. .. ..   on output commands, an
                    incorrect (FROM)LENGTH has
                    been specified, either less
                    than zero or greater than
                    32 767.
.. 08 .. .. .. ..   on input commands, an
                    incorrect (TO)LENGTH has
                    been specified, greater than
                    32 767.
.. 0C .. .. .. ..   length error has occurred on
                    ISSUE PASS command.
```

Figure 9. Byte 1 of EIBRCODE for LENGERR

Note: This field is not used exclusively, and can take other values.

6. When ILLOGIC or IOERR occurs during file control operations, further information is provided in field EIBRCODE, as shown in Figure 10. where:

```
.. xx xx xx xx ..   BDAM response.
.. xx .. .. .. ..   VSAM return code.
.. .. xx .. .. ..   VSAM error code.
```

Figure 10. EIBRCODE for ILLOGIC or IOERR

byte 3 =
VSAM problem determination code (ILLOGIC only)

byte 4 =
VSAM component code (ILLOGIC only)

Details of these response codes are described in the *DFSMS Macro Instructions for Data Sets* manual for VSAM, and the *DFSMS/MVS V1.3 Using Data Sets (SC26-4922)* for BDAM.

7. When INVMPSZ occurs during BMS operations, byte 3 of field EIBRCODE contains the terminal code as shown in Figure 11.

```
.. .. .. xx .. ..   terminal code.
```

Figure 11. Byte 3 of EIBRCODE for INVMPSZ

These are the same as the mapset suffixes shown in "DFHMSD" on page 877.

EIBRECV
Indicates that the application program is to continue receiving data from the facility by executing RECEIVE commands (X'FF').

```
COBOL:     PIC X(1).
C:         unsigned char eibrecv;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBREQID
Contains the request identifier assigned to an interval control command by CICS; this field is not used when a request identifier is specified in the

application program.

```
COBOL:     PIC X(8).
C:         char eibreqid [8];
PL/I:      CHAR(8)
Assembler: CL8
```

EIBRESP

Contains a number corresponding to the RESP condition that occurred. These numbers are listed (in decimal) for the conditions that can occur during execution of the commands described in this manual.

```
COBOL:     PIC S9(8) COMP
C:         long int eibresp;
PL/I:      FIXED BIN(31)
Assembler: F
```

No.	Condition	No.	Condition
00	NORMAL	60	SESSBUSY
01	ERROR	61	NOTALLOC
02	RDATT	62	CBIDERR
03	WRBRK	63	INVEXITREQ
04	EOF	64	INVPARTNSET
05	EODS	65	INVPARTN
06	EOC	66	PARTNFAIL
07	INBFMH	69	USERIDERR
08	ENDINPT	70	NOTAUTH
09	NONVAL	71	VOLIDERR
10	NOSTART	72	SUPPRESSED
11	TERMIDERR	75	RESIDERR
12	FILENOTFOUND	80	NOSPOOL
13	NOTFND	81	TERMERR
14	DUPREC	82	ROLLEDBACK
15	DUPKEY	83	END
16	INVREQ	84	DISABLED
17	IOERR	85	ALLOCERR
18	NOSPACE	86	STRELERR
19	NOTOPEN	87	OPENERR
20	ENDFILE	88	SPOLBUSY
21	ILLOGIC	89	SPOLERR
22	LENGERR	90	NODEIDERR
23	QZERO	91	TASKIDERR
24	SIGNAL	92	TCIDERR
25	QBUSY	93	DSNNOTFOUND
26	ITEMERR	94	LOADING
27	PGMIDERR	95	MODELIDERR
28	TRANSIDERR	96	OUTDESCRERR
29	ENDDATA	97	PARTNERIDERR
30	INVTSREQ	98	PROFILEIDERR
31	EXPIRED	99	NETNAMEIDERR
32	RETPAGE	100	LOCKED
33	RTEFAIL	101	RECORDBUSY
34	RTESOME	102	UOWNOTFOUND

No.	Condition	No.	Condition
35	TSIOERR	103	UOWLNOTFOUND
36	MAPFAIL	104	LINKABEND
37	INVERRTERM	105	CHANGED
38	INVMPSZ	106	PROCESSBUSY
39	IGREQID	107	ACTIVITYBUSY
40	OVERFLOW	108	PROCESSERR
41	INVLDC	109	ACTIVITYERR
42	NOSTG	110	CONTAINERERR
43	JIDERR	111	EVENTERR
44	QIDERR	112	TOKENERR
45	NOJBUFSP	113	NOTFINISHED
46	DSSTAT	114	POOLERR
47	SELNERR	115	TIMERERR
48	FUNCERR	116	SYMBOLERR
49	UNEXPIN	117	TEMPLATERR
50	NOPASSBKRD	118	NOTSUPERUSER
51	NOPASSBKWR	119	CSDERR
52	SEGIDERR	120	DUPRES
53	SYSIDERR	121	RESUNAVAIL
54	ISCINVREQ	122	CHANNELERR
55	ENQBUSY	123	CCSIDERR
56	ENVDEFERR	124	TIMEDOUT
57	IGREQCD	125	CODEPAGEERR
58	SESSIONERR	126	INCOMPLETE
59	SYSBUSY	127	APPNOTFOUND

EIBRESP2

Contains more detailed information that can help explain why the RESP condition occurred. This field contains meaningful values, as documented with each command to which it applies. For requests to remote files, EIBRESP2 contains binary zeros. If a program uses DPL to link to a program in another CICS region, an EIBRESP2 from the remote region is not returned to the program doing the DPL.

For programs written in C or C++, any value passed via the *exit* or *return* function is saved in EIBRESP2. This means that when DPL is used to link to a C or C++ program in a remote region, this value is not returned to the linking program.

```
COBOL:     PIC S9(8) COMP.
C:         long int eibresp2;
PL/I:      FIXED BIN(31)
Assembler: F
```

EIBRLDBK

Indicates rollback.

```
COBOL:     PIC X(1).
C:         unsigned char eibrldbk;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBRSRCE
Contains the symbolic identifier of the resource being accessed by the latest executed command as shown in Table 18

Table 18. Symbolic identifier of resource being accessed

Command type	Resource	Length
BMS	Map name	7
File control	File name	8
Interval control	Transaction name	4
Journal control	Journal number	H
Journal control	Journal name	8
Program control	Program name	8
Temporary storage control	TS queue name	8 or 16
Terminal control	Terminal name; LU name; LU6.1 session or APPC convid	4
Transient data control	TD queue name	4

Note:

1. H= halfword binary.
2. Identifiers less than eight characters in length are padded on the right with blanks.
3. Identifiers greater than eight characters in length are truncated.

```
COBOL:     PIC X(8).
C:         char eibrsrce [8];
PL/I:      CHAR(8)
Assembler: CL8
```

EIBSIG
Indicates that SIGNAL has been received (X'FF').

```
COBOL:     PIC X(1).
C:         unsigned char eibsig;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBSYNC
Indicates that the application program must take a sync point or terminate. Before either is done, the application program must ensure that any other facilities, owned by it, are put into the send state, or are freed (X'FF').

```
COBOL:     PIC X(1).
C:         unsigned char eibsync;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBSYNRB
Indicates that the application program should issue a SYNCPOINT ROLLBACK command (X'FF'). This field is only set in application programs holding a conversation on an APPC or MRO link.

```
COBOL:     PIC X(1).
C:         unsigned char eibsynrb;
PL/I:      CHAR(1)
Assembler: CL1
```

EIBTASKN
Contains the task number assigned to the task by CICS. This number appears in trace table entries generated while the task is in control. The format of the field is packed decimal.

```
COBOL:     PIC S9(7) COMP-3.
C:         char eibtaskn [4];
PL/I:      FIXED DEC(7,0)
Assembler: PL4
```

EIBTIME
Contains the time at which the task is started (this field is updated by the ASKTIME command). The time is in packed decimal form (0HHMMSS+), and can contain a value in the range 0000000+ to 0240000+. Both 0000000+ and 0240000+ are valid.

```
COBOL:     PIC S9(7) COMP-3.
C:         char eibtime [4];
PL/I:      FIXED DEC(7,0)
Assembler: PL4
```

EIBTRMID
Contains the symbolic terminal identifier of the principal facility (terminal or logical unit) associated with the task.

```
COBOL:     PIC X(4).
C:         char eibtrmid [4];
PL/I:      CHAR(4)
Assembler: CL4
```

The following prefixes are used to identify intercommunication sessions, terminals, and devices:

Table 19. Standard prefixes for sessions, terminals, and devices

Prefix	Session, terminal, or device
-	APPC session
}	Bridge facility
¬	Console
/	IPIC session
< or >	MRO session
{	Remote terminal
\ (default system initialization VTPREFIX value)	Virtual terminal

EIBTRNID
Contains the symbolic transaction identifier of the task.

```
COBOL:     PIC X(4).
C:         char eibtrnid [4];
PL/I:      CHAR(4)
Assembler: CL4
```

EXEC interface block (EIB) response and function codes

This appendix lists the response codes and the function codes of EXEC CICS commands.

Response codes of EXEC CICS commands

After the execution of an EXEC CICS command, fields EIBRESP and EIBRCODE are set to indicate whether the command executed successfully, or whether a CICS condition was raised. Use the response codes to help in problem determination.

Each possible value of EIBRESP relates directly to a specific condition, no matter which command caused the condition to be raised. This is not true for EIBRCODE values: both the value and the byte of EIBRCODE in which it is set depend on which command was issued.

The following sections list the conditions that are applicable to EXEC CICS commands, their corresponding RESP values (decimal), the associated EIBRCODE values (hexadecimal), and the transaction abend codes (if any).

EXEC CICS CSD, DISCARD, INQUIRE, PERFORM, and SET commands

The first word of EIBRCODE for these commands is always set equal to the hexadecimal equivalent of the RESP value; the remaining bytes are set to X'00'.

Condition (Byte 3)	RESP Value	EIBRCODE	Abend
BUSY	128	80	AEZ2
CSDERR	119	77	AEZS
DSNNOTFOUND	93	5D	AEX1
DUPREC	14	0E	AEIN
DUPRES	120	78	AEZT
END	83	53	AEXK

Condition (Byte 3)	RESP Value	EIBRCODE	Abend
FILENOTFOUND	12	0C	AEIL
ILLOGIC	21	15	AEIU
INCOMPLETE	126	7E	AEZZ
INVREQ	16	10	AEIP
IOERR	17	11	AEIQ
JIDERR	43	2B	AEYG
LENGERR	22	16	AEIV
LOCKED	100	64	AEX8
MODELIDERR	95	5F	AEX3
NOSPACE	18	12	AEIR
NOSTG	42	2A	—
NOTAUTH	70	46	AEY7
NOTFND	13	0D	AEIM
PARTNERIDERR	97	61	AEX5
PGMIDERR	27	1B	AEI0
PROFILEIDERR	98	62	AEX6
QIDERR	44	2C	AEYH
SYSBUSY	59	3B	—
SYSIDERR	53	35	AEYQ
TASKIDERR	91	5B	AEXX
TCIDERR	92	5C	AEX0
TERMIDERR	11	0B	AEIK
TRANSIDERR	28	1C	AEI1
UOWNOTFOUND	102	66	-
USERIDERR	69	45	AEYX
VOLIDERR	71	47	AEXV

EXEC CICS DISABLE, ENABLE, and EXTRACT EXIT commands

Conditions that can be raised by the DISABLE, ENABLE, and EXTRACT EXIT commands are INVEXITREQ and NOTAUTH. There are no conditions associated with the RESYNC command.

Condition	RESP Value	EIBRCODE	Abend
INVEXITREQ	63	80	AEY0
NOTAUTH	70	46	AEY7

Function codes of EXEC CICS commands

The function code (field EIBFN) is a hexadecimal value that identifies the command most recently issued by a task.

The format of the EIBFN field is as follows:

```
ASM     CL2
COBOL   PIC X(2)
PL/I    CHAR (2)
C       CHAR variable name(2);
```

The function codes of the EXEC CICS commands are listed first in command sequence, then in function code sequence.

Command	Code
ACQUIRE TERMINAL	86 02
COLLECT STATISTICS	70 08
CREATE ATOMSERVICE	30 42

Command	Code
CREATE BUNDLE	30 40
CREATE CONNECTION	30 0E
CREATE DB2CONN	30 20
CREATE DB2ENTRY	30 22
CREATE DB2TRAN	30 24
CREATE DOCTEMPLATE	30 2E
CREATE ENQMODEL	30 2A
CREATE FILE	30 14
CREATE IPCONN	30 3C
CREATE JOURNALMODEL	30 1E
CREATE JVMSERVER	30 46
CREATE LSRPOOL	30 16
CREATE LIBRARY	30 3E
CREATE MAPSET	30 04
CREATE MQCONN	30 44
CREATE PARTITIONSET	30 06
CREATE PARTNER	30 18
CREATE PIPELINE	30 38
CREATE PROCESSTYPE	30 26
CREATE PROFILE	30 0A
CREATE PROGRAM	30 02
CREATE SESSIONS	30 12
CREATE TCPIPSERVICE	30 30
CREATE TDQUEUE	30 1C
CREATE TERMINAL	30 10
CREATE TRANCLASS	30 1A
CREATE TRANSACTION	30 08
CREATE TSMODEL	30 28
CREATE TYPETERM	30 0C
CREATE URIMAP	30 36
CREATE WEBSERVICE	30 3A
CSD ADD	A2 02
CSD ALTER	A2 04
CSD APPEND	A2 06
CSD COPY	A2 08
CSD DEFINE	A2 0A
CSD DELETE	A2 0C
CSD DISCONNECT	A2 32
CSD ENDBRGROUP	A2 2C
CSD ENDBRLIST	A2 2E
CSD ENDBRRSRCE	A2 30
CSD GETNEXTGROUP	A2 20
CSD GETNEXTLIST	A2 22
CSD GETNEXTRSRCE	A2 24
CSD INQUIREGROUP	A2 1A
CSD INQUIRELIST	A2 1C
CSD INQUIRERSRCE	A2 1E
CSD INSTALL	A2 0E
CSD LOCK	A2 10
CSD REMOVE	A2 12
CSD RENAME	A2 14
CSD STARTBRGROUP	A2 26
CSD STARTBRLIST	A2 28
CSD STARTBRRSRCE	A2 2A

Command	Code
CSD UNLOCK	A2 16
CSD USERDEFINE	A2 18
DISABLE PROGRAM	22 04
DISCARD ATOMSERVICE	CC 10
DISCARD AUTINSTMODEL	42 10
DISCARD BUNDLE	C8 10
DISCARD CONNECTION	58 10
DISCARD DB2CONN	94 10
DISCARD DB2ENTRY	94 30
DISCARD DB2TRAN	94 50
DISCARD DOCTEMPLATE	9E 10
DISCARD ENQMODEL	90 90
DISCARD FILE	4C 10
DISCARD IPCONN	C2 10
DISCARD JOURNALMODEL	92 10
DISCARD JOURNALNAME	60 10
DISCARD JVMSERVER	B0 10
DISCARD LIBRARY	C6 10
DISCARD MQCONN	CE 10
DISCARD PARTNER	44 10
DISCARD PIPELINE	BC 10
DISCARD PROCESSTYPE	96 10
DISCARD PROFILE	46 10
DISCARD PROGRAM	4E 10
DISCARD TCPIPSERVICE	9C 10
DISCARD TDQUEUE	5C 10
DISCARD TERMINAL	52 10
DISCARD TRANCLASS	5E 18
DISCARD TRANSACTION	50 10
DISCARD TSMODEL	80 30
DISCARD URIMAP	BE 10
DISCARD WEBSERVICE	BC 30
ENABLE PROGRAM	22 02
EXTRACT EXIT	22 06
EXTRACT STATISTICS	70 26
INQUIRE ASSOCIATION	C4 02
INQUIRE ATOMSERVICE	CC 02
INQUIRE AUTINSTMODEL	42 02
INQUIRE AUTOINSTALL	68 12
INQUIRE BUNDLE	C8 02
INQUIRE BUNDLEPART	C8 12
INQUIRE BRFACILITY	B4 02
INQUIRE CAPDATAPRED	CA 2A
INQUIRE CAPINFOSRCE	CA 2C
INQUIRE CAPOPTPRED	CA 28
INQUIRE CAPTURESPEC	CA 22
INQUIRE CFDTPOOL	98 02
INQUIRE CONNECTION	58 02
INQUIRE DB2CONN	94 02
INQUIRE DB2ENTRY	94 22
INQUIRE DB2TRAN	94 42
INQUIRE DELETSHIPPED	68 22
INQUIRE DISPATCHER	B6 02
INQUIRE DOCTEMPLATE	9E 02

Command	Code
INQUIRE DSNAME	7A 02
INQUIRE DUMPDS	66 02
INQUIRE ENQ	90 22
INQUIRE ENQMODEL	90 82
INQUIRE EPADAPTER	CA 42
INQUIRE EPADAPTERSET	CA 32
INQUIRE EPADAPTINSET	CA 52
INQUIRE EVENTBINDING	CA 02
INQUIRE EVENTPROCESS	CA 12
INQUIRE EXCI	7C 02
INQUIRE EXITPROGRAM	88 02
INQUIRE FILE	4C 02
INQUIRE HOST	BE 12
INQUIRE IPCONN	C2 02
INQUIRE IPFACILITY	C2 12
INQUIRE IRC	6E 02
INQUIRE JOURNALMODEL	92 02
INQUIRE JOURNALNAME	60 12
INQUIRE JOURNALNUM	60 02
INQUIRE JVMSERVER	B0 42
INQUIRE LIBRARY	C6 02
INQUIRE MODENAME	5A 02
INQUIRE MONITOR	70 12
INQUIRE MQCONN	CE 02
INQUIRE MQINI	CE 12
INQUIRE MVSTCB	B6 12
INQUIRE NETNAME	52 16
INQUIRE NETNAME	52 06
INQUIRE OSGIBUNDLE	B0 52
INQUIRE OSGISERVICE	B0 62
INQUIRE PARTNER	44 02
INQUIRE PIPELINE	BC 02
INQUIRE PROCESSTYPE	96 02
INQUIRE PROFILE	46 02
INQUIRE PROGRAM	4E 02
INQUIRE REQID	8A 02
INQUIRE RRMS	3A 02
INQUIRE STATISTICS	70 02
INQUIRE STORAGE	5E 08
INQUIRE STREAMNAME	92 12
INQUIRE SUBPOOL	5E 42
INQUIRE SYSDUMPCODE	66 22
INQUIRE SYSTEM	54 02
INQUIRE TASK	5E 02
INQUIRE TCLASS	5E 12
INQUIRE TCPIP	9C 12
INQUIRE TCPIPSERVICE	9C 02
INQUIRE TDQUEUE	5C 02
INQUIRE TEMPSTORAGE	80 32
INQUIRE TERMINAL	52 02
INQUIRE TERMINAL	52 12
INQUIRE TRACEDEST	78 02
INQUIRE TRACEFLAG	78 12
INQUIRE TRACETYPE	78 22

Command	Code
INQUIRE TRANCLASS	5E 1A
INQUIRE TRANDUMPCODE	66 12
INQUIRE TRANSACTION	50 02
INQUIRE TSMODEL	80 22
INQUIRE TSPOOL	80 1A
INQUIRE TSQNAME	80 12
INQUIRE TSQUEUE	80 02
INQUIRE UOW	90 02
INQUIRE UOWDSNFAIL	90 62
INQUIRE UOWENQ	90 22
INQUIRE UOWLINK	90 42
INQUIRE URIMAP	BE 02
INQUIRE VOLUME	62 02
INQUIRE VTAM	68 02
INQUIRE WEB	9C 22
INQUIRE WEBSERVICE	BC 22
INQUIRE XMLTRANSFORM	D0 02
PERFORM DELETSHIPPED	68 26
PERFORM DUMP	7E 04
PERFORM ENDAFFINITY	58 06
PERFORM PIPELINE	BC 06
PERFORM RESETTIME	72 02
PERFORM SECURITY	64 02
PERFORM SHUTDOWN	76 02
PERFORM SSL	64 12
PERFORM STATISTICS	70 06
RESYNC ENTRYNAME	16 04
SET ATOMSERVICE	CC 04
SET AUTOINSTALL	68 14
SET BUNDLE	C8 04
SET BRFACILITY	B4 04
SET CONNECTION	58 04
SET DB2CONN	94 04
SET DB2ENTRY	94 24
SET DB2TRAN	94 44
SET DELETSHIPPED	68 24
SET DISPATCHER	B6 04
SET DSNAME	7A 04
SET DUMPDS	66 04
SET ENQMODEL	90 84
SET EPADAPTER	CA 44
SET EPADAPTERSET	CA 34
SET EVENTBINDING	CA 04
SET EVENTPROCESS	CA 14
SET FILE	4C 04
SET HOST	BE 14
SET IPCONN	C2 04
SET IRC	6E 04
SET JOURNALNAME	60 14
SET JOURNALNUM	60 04
SET JVMSERVER	B0 44
SET LIBRARY	C6 04
SET MODENAME	5A 04
SET MONITOR	70 14

Command	Code
SET MQCONN	CE 04
SET NETNAME	52 08
SET PIPELINE	BC 04
SET PROCESSTYPE	96 04
SET PROGRAM	4E 04
SET STATISTICS	70 04
SET SYSDUMPCODE	66 24
SET SYSTEM	54 04
SET TASK	5E 04
SET TCLASS	5E 14
SET TCPIP	9C 14
SET TCPIPSERVICE	9C 04
SET TDQUEUE	5C 04
SET TEMPSTORAGE	80 34
SET TERMINAL	52 04
SET TERMINAL	52 14
SET TRACEDEST	78 04
SET TRACEFLAG	78 14
SET TRACETYPE	78 24
SET TRANCLASS	5E 1C
SET TRANDUMPCODE	66 14
SET TRANSACTION	50 04
SET TSQNAME	80 14
SET TSQUEUE	80 04
SET UOW	90 04
SET UOWLINK	90 44
SET URIMAP	BE 04
SET VOLUME	62 04
SET VTAM	68 04
SET WEB	9C 24
SET WEBSERVICE	BC 24
SET XMLTRANSFORM	D0 04

Code	Command
16 04	RESYNC ENTRYNAME
22 02	ENABLE PROGRAM
22 04	DISABLE PROGRAM
22 06	EXTRACT EXIT
30 0A	CREATE PROFILE
30 0C	CREATE TYPETERM
30 0E	CREATE CONNECTION
30 02	CREATE PROGRAM
30 04	CREATE MAPSET
30 06	CREATE PARTITIONSET
30 08	CREATE TRANSACTION
30 1A	CREATE TRANCLASS
30 1C	CREATE TDQUEUE
30 1E	CREATE JOURNALMODEL
30 10	CREATE TERMINAL
30 12	CREATE SESSIONS
30 14	CREATE FILE
30 16	CREATE LSRPOOL
30 18	CREATE PARTNER

Code	Command
30 20	CREATE DB2CONN
30 22	CREATE DB2ENTRY
30 24	CREATE DB2TRAN
30 26	CREATE PROCESSTYPE
30 28	CREATE TSMODEL
30 2A	CREATE ENQMODEL
30 2C	CREATE REQUESTMODEL
30 2E	CREATE DOCTEMPLATE
30 30	CREATE TCPIPSERVICE
30 34	CREATE DJAR
30 36	CREATE URIMAP
30 38	CREATE PIPELINE
30 3A	CREATE WEBSERVICE
30 3C	CREATE IPCONN
30 3E	CREATE LIBRARY
30 40	CREATE BUNDLE
30 42	CREATE ATOMSERVICE
30 44	CREATE MQCONN
30 46	CREATE JVMSERVER
3A 02	INQUIRE RRMS
42 02	INQUIRE AUTINSTMODEL
42 10	DISCARD AUTINSTMODEL
44 02	INQUIRE PARTNER
44 10	DISCARD PARTNER
46 02	INQUIRE PROFILE
46 10	DISCARD PROFILE
4C 02	INQUIRE FILE
4C 04	SET FILE
4C 10	DISCARD FILE
4E 02	INQUIRE PROGRAM
4E 04	SET PROGRAM
4E 10	DISCARD PROGRAM
50 02	INQUIRE TRANSACTION
50 04	SET TRANSACTION
50 10	DISCARD TRANSACTION
52 02	INQUIRE TERMINAL
52 04	SET TERMINAL
52 06	INQUIRE NETNAME
52 08	SET NETNAME
52 10	DISCARD TERMINAL
52 12	INQUIRE TERMINAL
52 14	SET TERMINAL
52 16	INQUIRE NETNAME
54 02	INQUIRE SYSTEM
54 04	SET SYSTEM
58 02	INQUIRE CONNECTION
58 04	SET CONNECTION
58 06	PERFORM ENDAFFINITY
58 10	DISCARD CONNECTION
5A 02	INQUIRE MODENAME
5A 04	SET MODENAME
5C 02	INQUIRE TDQUEUE
5C 04	SET TDQUEUE
5C 10	DISCARD TDQUEUE

Code	Command
5E 02	INQUIRE TASK
5E 04	SET TASK
5E 08	INQUIRE STORAGE
5E 1A	INQUIRE TRANCLASS
5E 1C	SET TRANCLASS
5E 12	INQUIRE TCLASS
5E 14	SET TCLASS
5E 18	DISCARD TRANCLASS
5E 42	INQUIRE SUBPOOL
60 02	INQUIRE JOURNALNUM
60 04	SET JOURNALNUM
60 10	DISCARD JOURNALNAME
60 12	INQUIRE JOURNALNAME
60 14	SET JOURNALNAME
62 02	INQUIRE VOLUME
62 04	SET VOLUME
64 02	PERFORM SECURITY
64 12	PERFORM SSL
66 02	INQUIRE DUMPDS
66 04	SET DUMPDS
66 12	INQUIRE TRANDUMPCODE
66 14	SET TRANDUMPCODE
66 22	INQUIRE SYSDUMPCODE
66 24	SET SYSDUMPCODE
68 02	INQUIRE VTAM
68 04	SET VTAM
68 12	INQUIRE AUTOINSTALL
68 14	SET AUTOINSTALL
68 22	INQUIRE DELETSHIPPED
68 24	SET DELETSHIPPED
68 26	PERFORM DELETSHIPPED
6E 02	INQUIRE IRC
6E 04	SET IRC
70 02	INQUIRE STATISTICS
70 04	SET STATISTICS
70 06	PERFORM STATISTICS
70 08	COLLECT STATISTICS
70 12	INQUIRE MONITOR
70 14	SET MONITOR
70 26	EXTRACT STATISTICS
72 02	PERFORM RESETTIME
76 02	PERFORM SHUTDOWN
78 02	INQUIRE TRACEDEST
78 04	SET TRACEDEST
78 12	INQUIRE TRACEFLAG
78 14	SET TRACEFLAG
78 22	INQUIRE TRACETYPE
78 24	SET TRACETYPE
7A 02	INQUIRE DSNAME
7A 04	SET DSNAME
7C 02	INQUIRE EXCI
7E 04	PERFORM DUMP
80 02	INQUIRE TSQUEUE
80 04	SET TSQUEUE

Code	Command
80 12	INQUIRE TSQNAME
80 14	SET TSQNAME
80 1A	INQUIRE TSPOOL
80 22	INQUIRE TSMODEL
80 30	DISCARD TSMODEL
80 32	INQUIRE TEMPSTORAGE
80 34	SET TEMPSTORAGE
86 02	ACQUIRE TERMINAL
88 02	INQUIRE EXITPROGRAM
8A 02	INQUIRE REQID
90 02	INQUIRE UOW
90 04	SET UOW
90 22	INQUIRE ENQ
90 22	INQUIRE UOWENQ
90 42	INQUIRE UOWLINK
90 44	SET UOWLINK
90 62	INQUIRE UOWDSNFAIL
90 82	INQUIRE ENQMODEL
90 84	SET ENQMODEL
90 90	DISCARD ENQMODEL
92 02	INQUIRE JOURNALMODEL
92 10	DISCARD JOURNALMODEL
92 12	INQUIRE STREAMNAME
94 02	INQUIRE DB2CONN
94 04	SET DB2CONN
94 10	DISCARD DB2CONN
94 22	INQUIRE DB2ENTRY
94 24	SET DB2ENTRY
94 30	DISCARD DB2ENTRY
94 42	INQUIRE DB2TRAN
94 44	SET DB2TRAN
94 50	DISCARD DB2TRAN
96 02	INQUIRE PROCESSTYPE
96 04	SET PROCESSTYPE
96 10	DISCARD PROCESSTYPE
98 02	INQUIRE CFDTPOOL
9C 02	INQUIRE TCPIPSERVICE
9C 04	SET TCPIPSERVICE
9C 10	DISCARD TCPIPSERVICE
9C 12	INQUIRE TCPIP
9C 14	SET TCPIP
9C 22	INQUIRE WEB
9C 24	SET WEB
9E 02	INQUIRE DOCTEMPLATE
9E 10	DISCARD DOCTEMPLATE
A2 02	CSD ADD
A2 04	CSD ALTER
A2 06	CSD APPEND
A2 08	CSD COPY
A2 0A	CSD DEFINE
A2 0C	CSD DELETE
A2 0E	CSD INSTALL
A2 10	CSD LOCK
A2 12	CSD REMOVE

Code	Command
A2 14	CSD RENAME
A2 16	CSD UNLOCK
A2 18	CSD USERDEFINE
A2 1A	CSD INQUIREGROUP
A2 1C	CSD INQUIRELIST
A2 1E	CSD INQUIRERSRCE
A2 20	CSD GETNEXTGROUP
A2 22	CSD GETNEXTLIST
A2 24	CSD GETNEXTRSRCE
A2 26	CSD STARTBRGROUP
A2 28	CSD STARTBRLIST
A2 2A	CSD STARTBRRSRCE
A2 2C	CSD ENDBRGROUP
A2 2E	CSD ENDBRLIST
A2 30	CSD ENDBRRSRCE
A2 32	CSD DISCONNECT
B0 10	DISCARD JVMSERVER
B0 42	INQUIRE JVMSERVER
B0 44	SET JVMSERVER
B0 52	INQUIRE OSGIBUNDLE
B0 62	INQUIRE OSGISERVICE
B4 02	INQUIRE BRFACILITY
B4 04	SET BRFACILITY
B6 02	INQUIRE DISPATCHER
B6 04	SET DISPATCHER
B6 12	INQUIRE MVSTCB
BC 02	INQUIRE PIPELINE
BC 04	SET PIPELINE
BC 06	PERFORM PIPELINE
BC 10	DISCARD PIPELINE
BC 22	INQUIRE WEBSERVICE
BC 24	SET WEBSERVICE
BC 30	DISCARD WEBSERVICE
BE 02	INQUIRE URIMAP
BE 04	SET URIMAP
BE 10	DISCARD URIMAP
BE 12	INQUIRE HOST
BE 14	SET HOST
C2 02	INQUIRE IPCONN
C2 04	SET IPCONN
C2 10	DISCARD IPCONN
C2 12	INQUIRE IPFACILITY
C4 02	INQUIRE ASSOCIATION
C6 02	INQUIRE LIBRARY
C6 04	SET LIBRARY
C6 10	DISCARD LIBRARY
C8 02	INQUIRE BUNDLE
C8 04	SET BUNDLE
C8 12	INQUIRE BUNDLEPART
C8 10	DISCARD BUNDLE
CA 02	INQUIRE EVENTBINDING
CA 04	SET EVENTBINDING
CA 12	INQUIRE EVENTPROCESS
CA 14	SET EVENTPROCESS

Code	Command
CA 22	INQUIRE CAPTURESPEC
CA 28	INQUIRE CAPOPTPRED
CA 32	INQUIRE EPADAPTERSET
CA 34	SET EPADAPTERSET
CA 42	INQUIRE EPADAPTER
CA 44	SET EPADAPTER
CA 52	INQUIRE EPADAPTINSET
CA 2A	INQUIRE CAPDATAPRED
CA 2C	INQUIRE CAPINFOSRCE
CC 02	INQUIRE ATOMSERVICE
CC 04	SET ATOMSERVICE
CC 10	DISCARD ATOMSERVICE
CE 02	INQUIRE MQCONN
CE 04	SET MQCONN
CE 10	DISCARD MQCONN
CE 12	INQUIRE MQINI
D0 02	INQUIRE XMLTRANSFORM
D0 04	SET XMLTRANSFORM

Chapter 5. CICS-value data areas used by all commands

The CICS-value data-area (CVDA) values and their numeric equivalents for all of the EXEC CICS commands.

CVDAs are described in CICS-value data areas (CVDAs).

CVDAs and numeric values in alphabetic sequence

CVDA	Value
ABEND	900
ACQFAIL	515
ACQUIRED	69
ACQUIRING	71
ACTIVE	181
ACTIVITY	1002
ADD	291
ADDABLE	41
ADDFAIL	519
ADDRESS	895
ADVANCE	265
AINIT	282
ALARM	501
ALL	856
ALLCONN	169
ALLOCATD	81
ALLQUERY	431
ALLVALUES	860
ALTERABLE	52
ALTERNATE	197
ALTPRTCOPY	446
ANALYZER	1126
AND	1005
ANY	158
APLKYBD	391
APLTEXT	393
APPC	124
APPCPARALLEL	374
APPCSINGLE	373
APPEND	1036
APPLICATION	559

© Copyright IBM Corp. 1989, 2017

CVDA	Value
APPLID	303
APPLNAME	338
ASACTL	224
ASCII7	616
ASCII8	617
ASRUNTRAN	1209
ASSEMBLER	150
ASSERTED	1104
ASYNCHRONOUS	570
ATI	75
ATOM	1119
ATOMSERVICE	1179
ATTACH	576
ATTENTION	524
ATTLSAWARE	1205
AUDALARM	395
AUTOACTIVE	630
AUTOARCH	262
AUTOAUTH	1095
AUTOCONN	170
AUTOINACTIVE	631
AUTOINIT	284
AUTOINSTALL	1140
AUTOPAGE	80
AUTOREGISTER	1094
AUTOSTART	618
AUTOTERMID	1120
AUXILIARY	247
AUXPAUSE	313
AUXSTART	312
AUXSTOP	314
AVAILABLE	95
BACKOUT	192
BACKTRANS	397
BACKUPNONBWO	800
BASE	10
BASEAPI	1052
BASESPACE	664
BASE64	50
BASICAUTH	1092
BDAM	2

CVDA	Value
BEGINSESSION	510
BELOW	159
BGAM	63
BINARY	1038
BINIT	283
BIT	1020
BLK	47
BLOCKED	16
BOTH	1100
BRIDGE	935
BROWSABLE	39
BSAM	61
BTAM	62
BUNDLE	1180
BUSY	612
C	149
CACHE	791
CANCEL	526
CANCELLED	624
CANCELLING	1025
CAPTURESPEC	1195
CATEGORY	474
CBE	582
CBER	581
CCE	583
CD	491
CEDF	370
CERTIFICAUTH	1093
CERTUSER	1061
CFTABLE	833
CFE	584
CGROUP	902
CHANNEL	1047
CHAR	1019
CHUNKEND	737
CHUNKNO	735
CHUNKYES	736
CICS	660
CICSAPI	1052
CICSDATAKEY	379
CICSECURITY	195

CVDA	Value
CICSEXECKEY	381
CICSTABLE	101
CKOPEN	1055
CLASSCACHE	1075
CLEAR	640
CLICONVERT	743
CLIENT	1122
CLIENTAUTH	1032
CLIENTCERT	1087
CLOSE	741
CLOSED	19
CLOSEFAILED	349
CLOSELEAVE	261
CLOSEREQ	22
CLOSING	21
CLOUD	848
CMDPROT	673
CMDSECEXT	207
CMDSECNO	205
CMDSECYES	207
COBOL	151
COBOLII	375
COLD	788
COLDACQ	72
COLDQUERY	433
COLDSTART	266
COLLECTION	473
COLOR	399
COMMAREA	1048
COMMIT	208
COMMITFAIL	792
COMPAT	107
COMPLETE	1026
COMPOSITE	1003
COMPRESS	1130
CONFFREE	82
CONFRECV	83
CONFSEND	84
CONNECT	903
CONNECTED	690
CONNECTING	904

CVDA	Value
CONNECTION	755
CONSISTENT	723
CONSOLE	66
CONTAINER	869
CONTENTION	836
CONTEXT	574
CONTROLSHUT	623
CONVERSE	600
CONVIDLE	518
COORDINATOR	770
COPID	908
COPY	401
CORBA	1101
CORBASERVER	1141
CPLUSPLUS	148
CREATE	67
CREATESPI	844
CRITICAL	11
CSDAPI	840
CSDBATCH	841
CSIGN	905
CTERM	906
CTLGALL	632
CTLGMODIFY	633
CTLGNONE	634
CTRLABLE	56
CTX	907
CURRENT	260
CURRENTPGM	870
CUSERID	909
CUSTOM	579
DAEOPT	684
DATA	508
DATASET	756
DATASETFULL	793
DATASTREAM	543
DB2CONN	1142
DB2ENTRY	1143
DB2TRAN	1184
DEADLOCK	794
DEBUG	1082

CVDA	Value
DEC	46
DEFAULT	198
DEFAULTUSER	1070
DEFINITION	990
DEFRESP1	497
DEFRESP1OR2	528
DEFRESP2	498
DEFRESP3	499
DELAY	637
DELETABLE	43
DELETE	292
DELETEFAIL	520
DELEXITERROR	795
DEREGERROR	679
DEREGISTERED	678
DEST	235
DISABLED	24
DISABLEDHOST	1125
DISABLING	25
DISCARDFAIL	513
DISCARDING	1118
DISCONNING	910
DISCREQ	444
DISK1	252
DISK2	253
DISK2PAUSE	254
DISPATCHABLE	228
DISPATCHER	1144
DJAR	1189
DOCDELETE	295
DOCTEMPLATE	1145
DOESNOTEQUAL	861
DOESNOTEXIST	872
DOESNOTSTART	862
DORMANT	1024
DPLSUBSET	383
DRAINING	619
DREPAPI	843
DREPBATCH	845
DS3270	615
DSIE	1206

CVDA	Value
DUALCASE	403
DUMMY	781
DUPERROR	106
DUPNOREPLACE	105
DUPREPLACE	104
DYNAMIC	178
EB	490
EBCDIC	1039
ECI	1098
EJB	1102
EMERGENCY	268
EMPTY	210
EMPTYREQ	31
ENABLED	23
ENABLING	51
ENDAFFINITY	790
ENQMODEL	1188
ENQUEUE	1146
ENTRYPOINT	993
EPADAPTER	1196
EPADAPTERSET	1200
EQUAL	911
EQUALS	863
ESDS	5
EVENT	334
EVENTBINDING	1191
EVENTPROCESS	1192
EVENTUAL	3
EXCEPT	332
EXCEPTRESP	523
EXCI	650
EXCTL	48
EXECENQ	751
EXECENQADDR	752
EXECINSTALL	846
EXISTS	871
EXIT	912
EXITTRACE	362
EXPECT	747
EXPIRED	1017
EXPORT	991

CVDA	Value
EXTENDED	1128
EXTENDEDDS	405
EXTRA	221
EXTSECURITY	194
FAILED	782
FAILEDBKOUT	357
FAILINGBKOUT	358
FAULTTOEPR	853
FCLOSE	273
FEED	472
FILE	238
FINALQ	183
FINPUT	270
FIRED	1001
FIRSTINIT	625
FIRSTQ	182
FIXED	12
FLUSH	783
FMH	502
FMHPARM	385
FOPEN	272
FORCE	342
FORCECANCEL	655
FORCECLOSE	351
FORCED	1013
FORCEPURGE	237
FORCLOSING	353
FORMATEDF	606
FORMATTED	542
FORMFEED	407
FOUTPUT	271
FREE	85
FREEING	94
FROMEPR	850
FULL	212
FULLAPI	384
FULLAUTO	1073
FWDRECOVABLE	354
GENERIC	651
GET	748
GMT	604

CVDA	Value
GOHIGHERTHAN	873
GOINGOUT	172
GOLOWERTHAN	874
GREATERTHAN	864
GROUP	913
GROUPRESYNC	947
GRPLIST	842
GTFSTART	317
GTFSTOP	318
HEAD	750
HEURBACKOUT	758
HEURCOMMIT	757
HEX	45
HFORM	409
HFSFILE	1040
HIGH	914
HILIGHT	413
HOLD	163
HOME	1103
HOSTNAME	302
HTTP	1096
HTTPNO	1034
HTTPS	1029
HTTPYES	1033
IDENTIFY	1074
IDNTY	328
IGNORE	1
IGNORERR	816
IIOP	1097
IMMCLOSE	350
IMMCLOSING	352
IMMEDIATE	2
IMMQUIESCED	706
IMPORT	992
INACTIVE	378
INBOUND	547
INCOMPLETE	1014
INDEXRECFULL	796
INDIRECT	122
INDOUBT	620
INFLIGHT	621

CVDA	Value
INITCOMPLETE	628
INITIAL	789
INITING	1112
INOUT	532
INPUT	226
INSERVICE	73
INSTALLED	550
INSTALLFAIL	512
INTERNAL	1058
INTRA	222
INTSTART	310
INTSTOP	311
INVALID	359
IOERROR	797
IPCONN	1176
IPECI	936
IPIC	1089
IPV4	300
IPV6	301
IRC	121
ISNOTGREATER	865
ISNOTLESS	866
ISOLATE	658
JAVA	147
JOURNALMODEL	1185
JOURNALNAME	1147
JOURNALNUM	1148
JVM	1080
JVMPOOL	1149
JVMPROFILE	1150
JVMPROGRAM	1151
JVMSERVER	1193
KATAKANA	415
KEEP	786
KERBEROS	1105
KEYED	8
KILL	232
KSDS	6
LCKSTRUCFULL	832
LEAVE	811
LESSTHAN	867

CVDA	Value
LE370	377
LIBRARY	1177
LIC	493
LIGHTPEN	417
LINK	577
LOAD	834
LOC24	159
LOC31	158
LOCAL	605
LOCKING	837
LOG	54
LOGICAL	216
LOGREPLICATE	1198
LOGTERM	269
LOSE	544
LOW	915
LPA	165
LSRPOOL	1152
LUP	541
LUSTAT	525
LUW	246
LU61	125
MAIN	248
MAP	155
MAPSET	155
MCHCTL	241
MDT	506
METADATA	857
MESSAGEID	1197
MIXED	403
MNPS	999
MOD	813
MODEANY	189
MODEL	370
MODE24	190
MODE31	191
MONITOR	1153
MORE	492
MQCONN	1175
MRO	938
MSRCONTROL	419

CVDA	Value
MTOM	460
MTOMNOXOP	469
MVS	780
MVSTCB	1154
NEGATIVE	530
NEW	28
NEWCOPY	167
NEWSESSION	485
NOALARM	500
NOALTPRTCOPY	447
NOANALYZER	1127
NOAPLKYBD	392
NOAPLTEXT	394
NOAPPEND	1037
NOAPPLNAME	339
NOATI	76
NOAUDALARM	396
NOAUTHENTIC	1091
NOAUTO	1071
NOAUTOARCH	263
NOAUTOSTART	1208
NOBACKOUT	193
NOBACKTRANS	398
NOCEDF	371
NOCLASSCACHE	1076
NOCLEAR	641
NOCLICONVERT	744
NOCLOSE	742
NOCMDPROT	674
NOCOLOR	400
NOCOMPAT	108
NOCOMPRESS	1131
NOCONNECT	916
NOCONV	556
NOCONVERSE	601
NOCONVERT	734
NOCOPY	402
NOCREATE	68
NOCTL	223
NODAEOPT	685
NODE	1155

CVDA	Value
NODEBUG	1083
NODISCREQ	445
NODOCDELETE	296
NODUALCASE	404
NOEMPTYREQ	32
NOEVENT	335
NOEXCEPT	333
NOEXCTL	49
NOEXIT	917
NOEXITTRACE	363
NOEXTENDEDDS	406
NOFMH	503
NOFMHPARM	386
NOFORCE	1054
NOFORMATEDF	607
NOFORMFEED	408
NOHFORM	410
NOHILIGHT	414
NOHOLD	164
NOIDNTY	329
NOINCONVERT	745
NOISOLATE	657
NOJVM	1081
NOKATAKANA	416
NOKEEP	787
NOLIGHTPEN	418
NOLOAD	835
NOLOG	55
NOLOGREPLICA	1199
NOLOSTLOCKS	710
NOMDT	507
NOMSGJRNL	531
NOMSRCONTROL	420
NOMTOM	461
NOMTOMNOXOP	470
NONAUTOCONN	171
NONCICS	661
NONCRITICAL	1090
NONE	496
NONLE370	1084
NONRESIDENT	1202

CVDA	Value
NONTRANS	572
NOOBFORMAT	422
NOOBOPERID	388
NOOUTCONVERT	746
NOOUTLINE	424
NOPARTITIONS	426
NOPERF	331
NOPHASEOUT	1110
NOPRESETSEC	243
NOPRINTADAPT	428
NOPROGSYMBOL	430
NOPRTCOPY	449
NOPS	997
NOQUERY	432
NORECOVDATA	700
NOREENTPROT	681
NORELEASE	918
NORELREQ	443
NOREPLICATOR	1212
NORESRCE	337
NORESYNC	934
NORETAINED	711
NOREUSE	294
NORMAL	1016
NORMALBKOUT	356
NORMALRESP	522
NORMI	775
NOROLLBACK	919
NOSECURITY	196
NOSENDMTOM	464
NOSHUTDOWN	289
NOSOSI	435
NOSPI	694
NOSRVCONVERT	739
NOSSL	1031
NOSTSN	487
NOSWITCH	285
NOSYNCPOINT	603
NOSYSDUMP	185
NOSYSLOG	784
NOTADDABLE	42

CVDA	Value
NOTALLOWED	327
NOTALTERABLE	53
NOTAPPLIC	1
NOTASKSTART	608
NOTBRWBLE	40
NOTBUSY	613
NOTCDEB	495
NOTCONNECTED	691
NOTCTRLABLE	57
NOTDEFINED	659
NOTDELBLE	44
NOTDYNAMIC	1021
NOTEMPTY	211
NOTEXTENDED	1129
NOTERMINAL	214
NOTEXTKYBD	437
NOTEXTPRINT	439
NOTFIRED	1000
NOTFWDRCVBLE	361
NOTHOTPOOL	1066
NOTINBOUND	546
NOTINIT	376
NOTINSTALLED	551
NOTKEYED	9
NOTLPA	166
NOTPENDING	127
NOTPURGEABLE	161
NOTRANDUMP	187
NOTREADABLE	36
NOTREADY	259
NOTRECOVABLE	30
NOTREQUIRED	667
NOTRLS	721
NOTROUTABLE	1022
NOTSOS	669
NOTSUPPORTED	15
NOTSUSPENDED	1027
NOTTABLE	100
NOTTI	78
NOTUPDBLE	38
NOTWAIT	920

CVDA	Value
NOUCTRAN	451
NOUSER	1011
NOVALIDATION	441
NOVFORM	412
NOWAIT	341
NOWRITE	275
NOXOPDIRECT	466
NOXOPSUPPORT	468
NOZCPTRACE	365
NRS	774
N906	931
N906D	921
OBFORMAT	421
OBOPERID	387
OBTAINING	96
OFF	200
OK	274
OLD	26
OLDCOPY	162
OLDSESSION	486
ON	201
OPEN	18
OPENAPI	1053
OPENERROR	798
OPENING	20
OPENINPUT	256
OPENOUTPUT	257
OPID	933
OPTIONAL	326
OPTIONS	759
OR	1006
OTS	1107
OUTLINE	423
OUTPUT	227
OUTSERVICE	74
OWNER	753
PAGEABLE	79
PARTITIONS	425
PARTITIONSET	156
PARTNER	1182
PATH	11

CVDA	Value
PDSMEMBER	1035
PENDBEGIN	558
PENDDATA	560
PENDFREE	86
PENDING	126
PENDINIT	1113
PENDPASS	565
PENDRECEIVE	87
PENDRELEASE	562
PENDRESOLVE	1114
PENDSTART	561
PENDSTSN	557
PENDUNSOL	564
PERF	330
PERMANENT	484
PHASEIN	168
PHASEOUT	1111
PHYSICAL	215
PIPELINE	1124
PLI	152
PL1	152
POLICYSCOPE	994
POOL	922
POSITIVE	529
POST	636
POSTCOMMAND	697
PRECOMMAND	696
PRESETSEC	242
PRIMARY	110
PRINCIPAL	199
PRINTADAPT	427
PRIVATE	174
PROCESS	1010
PROCESSTYPE	1190
PROFILE	157
PROGAUTO	1072
PROGRAM	154
PROGRAMDEF	1178
PROGRAMINIT	698
PROGSYMBOL	429
PROTECTED	504

CVDA	Value
PROVIDER	1041
PRTCOPY	448
PUBLISH	1085
PURGE	236
PURGEABLE	160
PUT	749
QR	1057
QUASIRENT	1050
QUEUE	814
QUIESCED	707
QUIESCING	708
READABLE	35
READBACK	209
READONLY	275
READY	258
REBUILD	923
RECEIVE	88
RECEIVER	1046
RECONNECT	924
RECOVDATA	701
RECOVERABLE	29
RECOVERED	277
RECOVERLOCKS	712
RECOVERY	1156
REENTPROT	680
REFPARMS	858
REGERROR	677
REGION	575
REGISTERED	670
REJECT	815
RELATED	675
RELEASE	563
RELEASED	70
RELEASING	549
RELOAD	1078
RELOADING	1079
RELREQ	442
REMLOSTLOCKS	713
REMOTE	4
REMOVE	276
REMSESSION	740

CVDA	Value
REMTABLE	103
REPEATABLE	724
REPLICATOR	1211
REPLYTOEPR	852
REQCONTEXT	854
REQUEST	249
REQUESTER	1042
REQUESTMODEL	1157
REQUIRED	666
REREAD	812
RESET	290
RESETLOCKS	714
RESIDENT	1201
RESOLVED	347
RESOLVING	1115
RESRCE	336
RESSECEXT	204
RESSECINT	203
RESSECNO	202
RESSECYES	204
RESPCONTEXT	855
RESPECTED	1109
RESYNC	702
RETAINED	715
RETRACT	1086
RETRY	716
REUSE	293
REVERTED	264
RFC1123	646
RFC3339	647
RLS	720
RLSACTIVE	730
RLSGONE	799
RLSINACTIVE	731
RLSSERVER	761
RMI	771
ROLLBACK	89
ROUTABLE	1023
ROUTE	638
RPG	153
RRCOMMITFAIL	830

CVDA	Value
RRDS	7
RREPL	1210
RRINDOUBT	831
RRMS	1063
RRSUR	939
RTR	527
RU	494
RUNNING	229
RZINSTOR	940
SAMESENDMTOM	462
SCAN	1088
SCHEDULER	941
SCS	614
SECERROR	1214
SECONDINIT	626
SECUSER	1060
SEND	90
SENDER	1045
SENDMTOM	463
SERVER	1123
SERVICE	475
SESSION	372
SESSIONFAIL	517
SESSIONLOST	516
SESSIONS	1187
SETFAIL	514
SHARE	27
SHARED	173
SHUNTED	762
SHUTDISABLED	645
SHUTDOWN	288
SHUTENABLED	644
SIGN	925
SIGNEDOFF	245
SIGNEDON	244
SINGLEOFF	324
SINGLEON	323
SKIP	810
SMF	255
SNPS	998
SOAP	1044

CVDA	Value
SOCKET	942
SOMEAVAIL	1204
SOS	668
SOSABOVE	683
SOSBELOW	682
SOSI	434
SPECIAL	98
SPECIFIC	652
SPECTRACE	177
SPI	693
SPRSTRACE	175
SQLCODE	926
SRVCONVERT	738
SSL	1030
STANDARD	97
STANDBY	629
STANTRACE	176
START	635
STARTED	609
STARTING	772
STARTTERM	943
STARTING	772
STARTSWITH	868
STARTUP	180
STATIC	179
STATS	1158
STOPPED	610
STOPPING	348
STORAGE	1159
STREAMNAME	1160
STSN	509
STSNSET	488
STSNTEST	489
SUBORDINATE	773
SUBPOOL	1161
SUBSPACE	663
SUMMUNLIKE	566
SUPPORTED	1106
SURROGATE	371
SUSPENDED	231
SWITCH	188

CVDA	Value
SWITCHALL	287
SWITCHING	225
SWITCHNEXT	286
SYNCFREE	91
SYNCHRONOUS	571
SYNCPOINT	602
SYNCRECV	92
SYNCSEND	93
SYSCONNECT	653
SYSDUMP	184
SYSDUMPCODE	1162
SYSLOG	785
SYSPLEX	656
SYSTEM	643
SYSTEMOFF	320
SYSTEMON	319
TABLE	847
TABLEMGR	1163
TAKEOVER	111
TAPE1	250
TAPE2	251
TASK	233
TASKSUBPOOL	1164
TASKSTART	611
TCAM	64
TCAMSNA	65
TCEXITALL	366
TCEXITALLOFF	369
TCEXITNONE	368
TCEXITSYSTEM	367
TCLASS	239
TCPIP	802
TCPIPSERVICE	1166
TDQ	767
TDQUEUE	767
TEMPLATENAME	1168
TEMPORARY	483
TERM	234
TERMINAL	213
TERMINATE	1077
TEXTKYBD	436

CVDA	Value
TEXTPRINT	438
THIRDINIT	627
THREADSAFE	1051
TIME	240
TIMEOUT	511
TIMER	1004
TOEPR	851
TPOOL	932
TPS55M2	552
TPS55M3	553
TPS55M4	554
TPS55M5	555
TRACE	760
TRANCLASS	1169
TRANDATA	944
TRANDUMP	186
TRANDUMPCODE	1170
TRANIDONLY	452
TRANS	573
TRANSACTION	1171
TRANSTART	580
TSMODEL	1183
TSQ	768
TSQUEUE	768
TTI	77
TWAIT	927
TX	929
TXID	928
TYPETERM	1186
T3278M2	533
T3278M3	534
T3278M4	535
T3278M5	536
T3279M2	537
T3279M3	538
T3279M4	539
T3279M5	540
UCTRAN	450
UKOPEN	1056
UNATTEMPTED	820
UNAVAILABLE	672

CVDA	Value
UNBLOCKED	17
UNCOMMITTED	722
UNCONNECTED	703
UNDEFINED	14
UNDETERMINED	355
UNENABLED	33
UNENABLING	34
UNEXPECTED	1015
UNEXPIRED	1018
UNINSTALLED	548
UNKNOWN	774
UNPROTECTED	505
UNQUIESCED	709
UNREGISTERED	671
UNRESOLVED	1116
UNSOLDATA	521
UNUSABLE	1117
UOW	246
UPDATABLE	37
UPDATING	1049
URIMAP	1173
URMTERMID	1121
USER	642
USERDATAKEY	380
USEREXECKEY	382
USERID	930
USEROFF	322
USERON	321
USERTABLE	102
UTC	1203
VALID	360
VALIDATION	440
VARIABLE	13
VERIFY	1099
VFORM	411
VRRDS	732
VSAM	3
VTAM	60
WAIT	340
WAITCOMMIT	763
WAITER	754

CVDA	Value
WAITFORGET	622
WAITING	765
WAITRMI	766
WAITRRMS	1063
WARMSTART	267
WBE	585
WEB	945
WEBSERVICE	1174
WIN	545
WMQ	578
XCF	665
XM	123
XMLTRANSFORM	1194
XMRUNTRAN	946
XNOTDONE	144
XOK	143
XOPDIRECT	465
XOPSUPPORT	467
XPLINK	1068
ZCPTRACE	364

Note: VTAM is now z/OS Communications Server.

CVDA values for the DEVICE option

CVDA sequence	
BATCHLU	191
BIPROG	160
BISYNCH	128
CDRDLPRT	24
CONTNLU	189
HARDCOPY	32
INTACTLU	190
ISCMMCONV	209
LUCMODGRP	210
LUCSESS	211
LUTYPE4	193
LUTYPE6	192
MAGTAPE	20
RESSYS	208
SDLC	176

CVDA sequence	
SEQDISK	18
SYSTEM3	161
SYSTEM7	2
SYS370	164
SYS7BSCA	166
TCONSOLE	8
TELETYPE	34
TTCAM	80
TWX3335	33
T1050	36
T1053	74
T2260L	65
T2260R	72
T2265	76
T2740	40
T2741BCD	43
T2741COR	42
T2770	130
T2780	132
T2980	134
T3275R	146
T3277L	153
T3277R	145
T3284L	155
T3284R	147
T3286L	156
T3286R	148
T3600BI	138
T3601	177
T3614	178
T3650ATT	186
T3650PIPE	184
T3650USER	187
T3653HOST	185
T3735	136
T3740	137
T3780	133
T3790	180
T3790SCSP	182
T3790UP	181
T7770	1

CVDA sequence	
VIDEOTERM	64

Numeric sequence	
1	T7770
2	SYSTEM7
8	TCONSOLE
18	SEQDISK
20	MAGTAPE
24	CDRDLPRT
32	HARDCOPY
33	TWX3335
34	TELETYPE
36	T1050
40	T2740
42	T2741COR
43	T2741BCD
64	VIDEOTERM
65	T2260L
72	T2260R
74	T1053
76	T2265
80	TTCAM
128	BISYNCH
130	T2770
132	T2780
133	T3780
134	T2980
136	T3735
137	T3740
138	T3600BI
145	T3277R
146	T3275R
147	T3284R
148	T3286R
153	T3277L
155	T3284L
156	T3286L
160	BIPROG
161	SYSTEM3
164	SYS370
166	SYS7BSCA

Numeric sequence	
176	SDLC
177	T3601
178	T3614
180	T3790
181	T3790UP
182	T3790SCSP
184	T3650PIPE
185	T3653HOST
186	T3650ATT
187	T3650USER
189	CONTNLU
190	INTACTLU
191	BATCHLU
192	LUTYPE6
193	LUTYPE4
208	RESSYS
209	ISCMMCONV
210	LUCMODGRP
211	LUCSESS

CVDAs and numeric values in numeric sequence

Value	CVDA
1	IGNORE
1	NOTAPPLIC
2	BDAM
2	IMMEDIATE
3	EVENTUAL
3	VSAM
4	REMOTE
5	ESDS
6	KSDS
7	RRDS
8	KEYED
9	NOTKEYED
10	BASE
11	CRITICAL
11	PATH
12	FIXED
13	VARIABLE
14	UNDEFINED

Value	CVDA
15	NOTSUPPORTED
16	BLOCKED
17	UNBLOCKED
18	OPEN
19	CLOSED
20	OPENING
21	CLOSING
22	CLOSEREQUEST
23	ENABLED
24	DISABLED
25	DISABLING
26	OLD
27	SHARE
28	NEW
29	RECOVERABLE
30	NOTRECOVABLE
31	EMPTYREQ
32	NOEMPTYREQ
33	UNENABLED
34	UNENABLING
35	READABLE
36	NOTREADABLE
37	UPDATABLE
38	NOTUPDATABLE
39	BROWSABLE
40	NOTBROWSABLE
41	ADDABLE
42	NOTADDABLE
43	DELETABLE
44	NOTDELETABLE
45	HEX
46	DEC
47	BLK
48	EXCTL
49	NOEXCTL
50	BASE64
51	ENABLING
52	ALTERABLE
53	NOTALTERABLE
54	LOG
55	NOLOG

Value	CVDA
56	CTRLABLE
57	NOTCTRLABLE
60	VTAM
61	BSAM
62	BTAM
63	BGAM
64	TCAM
65	TCAMSNA
66	CONSOLE
67	CREATE
68	NOCREATE
69	ACQUIRED
70	RELEASED
71	ACQUIRING
72	COLDACQ
73	INSERVICE
74	OUTSERVICE
75	ATI
76	NOATI
77	TTI
78	NOTTI
79	PAGEABLE
80	AUTOPAGEABLE
81	ALLOCATED
82	CONFFREE
83	CONFRECEIVE
84	CONFSEND
85	FREE
86	PENDFREE
87	PENDRECEIVE
88	RECEIVE
89	ROLLBACK
90	SEND
91	SYNCFREE
92	SYNCRECEIVE
93	SYNCSEND
94	FREEING
95	AVAILABLE
96	OBTAINING
97	STANDARD
98	SPECIAL

Value	CVDA
100	NOTTABLE
101	CICSTABLE
102	USERTABLE
103	REMTABLE
110	PRIMARY
111	TAKEOVER
121	IRC
122	INDIRECT
123	XM
124	APPC
125	LU61
126	PENDING
127	NOTPENDING
143	XOK
144	XNOTDONE
147	JAVA
148	CPLUSPLUS
149	C
150	ASSEMBLER
151	COBOL
152	PLI
152	PL1
153	RPG
154	PROGRAM
155	MAP
155	MAPSET
156	PARTITIONSET
157	PROFILE
158	ANY
158	LOC31
159	BELOW
159	LOC24
160	PURGEABLE
161	NOTPURGEABLE
162	OLDCOPY
163	HOLD
164	NOHOLD
165	LPA
166	NOTLPA
167	NEWCOPY
168	PHASEIN

Value	CVDA
169	ALLCONN
170	AUTOCONN
171	NONAUTOCONN
172	GOINGOUT
173	SHARED
174	PRIVATE
175	SPRSTRACE
176	STANTRACE
177	SPECTRACE
178	DYNAMIC
179	STATIC
180	STARTUP
181	ACTIVE
182	FIRSTQUIESCE
183	FINALQUIESCE
184	SYSDUMP
185	NOSYSDUMP
186	TRANDUMP
187	NOTRANDUMP
188	SWITCH
189	MODEANY
190	MODE24
191	MODE31
192	BACKOUT
193	NOBACKOUT
194	EXTSECURITY
195	CICSECURITY
196	NOSECURITY
197	ALTERNATE
198	DEFAULT
199	PRINCIPAL
200	OFF
201	ON
202	RESSECNO
203	RESSECINT
204	RESSECEXT
204	RESSECYES
205	CMDSECNO
207	CMDSECEXT
207	CMDSECYES
208	COMMIT

Value	CVDA
209	READBACK
210	EMPTY
211	NOTEMPTY
212	FULL
213	TERMINAL
214	NOTERMINAL
215	PHYSICAL
216	LOGICAL
221	EXTRA
222	INTRA
223	NOCTL
224	ASACTL
225	SWITCHING
226	INPUT
227	OUTPUT
228	DISPATCHABLE
229	RUNNING
231	SUSPENDED
232	KILL
233	TASK
234	TERM
235	DEST
236	PURGE
237	FORCEPURGE
238	FILE
239	TCLASS
240	TIME
241	MCHCTL
242	PRESETSEC
243	NOPRESETSEC
244	SIGNEDON
245	SIGNEDOFF
246	LUW
247	AUXILIARY
248	MAIN
249	REQUEST
250	TAPE1
251	TAPE2
252	DISK1
253	DISK2
254	DISK2PAUSE

Value	CVDA
255	SMF
256	OPENINPUT
257	OPENOUTPUT
258	READY
259	NOTREADY
260	CURRENT
261	CLOSELEAVE
262	AUTOARCH
263	NOAUTOARCH
264	REVERTED
265	ADVANCE
266	COLDSTART
267	WARMSTART
268	EMERGENCY
269	LOGTERM
270	FINPUT
271	FOUTPUT
272	FOPEN
273	FCLOSE
274	OK
275	NOWRITE
275	READONLY
276	REMOVE
277	RECOVERED
282	AINIT
283	BINIT
284	AUTOINIT
285	NOSWITCH
286	SWITCHNEXT
287	SWITCHALL
288	SHUTDOWN
289	NOSHUTDOWN
290	RESET
291	ADD
292	DELETE
293	REUSE
294	NOREUSE
300	IPV4
301	IPV6
302	HOSTNAME
303	APPLID

Value	CVDA
310	INTSTART
311	INTSTOP
312	AUXSTART
313	AUXPAUSE
314	AUXSTOP
317	GTFSTART
318	GTFSTOP
319	SYSTEMON
320	SYSTEMOFF
321	USERON
322	USEROFF
323	SINGLEON
324	SINGLEOFF
326	OPTIONAL
327	NOTALLOWED
328	IDNTY
329	NOIDNTY
330	PERF
331	NOPERF
332	EXCEPT
333	NOEXCEPT
334	EVENT
335	NOEVENT
336	RESRCE
337	NORESRCE
338	APPLNAME
339	NOAPPLNAME
340	WAIT
341	NOWAIT
347	RESOLVED
348	STOPPING
349	CLOSEFAILED
350	IMMCLOSE
351	FORCECLOSE
352	IMMCLOSING
353	FORCECLOSING
354	FWDRECOVABLE
355	UNDETERMINED
356	NORMALBKOUT
357	FAILEDBKOUT
358	FAILINGBKOUT

Value	CVDA
359	INVALID
360	VALID
361	NOTFWDRCVBLE
362	EXITTRACE
363	NOEXITTRACE
364	ZCPTRACE
365	NOZCPTRACE
366	TCEXITALL
367	TCEXITSYSTEM
368	TCEXITNONE
369	TCEXITALLOFF
370	CEDF
370	MODEL
371	NOCEDF
371	SURROGATE
372	SESSION
373	APPCSINGLE
374	APPCPARALLEL
375	COBOLII
376	NOTINIT
377	LE370
378	INACTIVE
379	CICSDATAKEY
380	USERDATAKEY
381	CICSEXECKEY
382	USEREXECKEY
383	DPLSUBSET
384	FULLAPI
385	FMHPARM
386	NOFMHPARM
387	OBOPERID
388	NOOBOPERID
391	APLKYBD
392	NOAPLKYBD
393	APLTEXT
394	NOAPLTEXT
395	AUDALARM
396	NOAUDALARM
397	BACKTRANS
398	NOBACKTRANS
399	COLOR

Value	CVDA
400	NOCOLOR
401	COPY
402	NOCOPY
403	MIXED
404	NODUALCASE
405	EXTENDEDDS
406	NOEXTENDEDDS
407	FORMFEED
408	NOFORMFEED
409	HFORM
410	NOHFORM
411	VFORM
412	NOVFORM
413	HILIGHT
414	NOHILIGHT
415	KATAKANA
416	NOKATAKANA
417	LIGHTPEN
418	NOLIGHTPEN
419	MSRCONTROL
420	NOMSRCONTROL
421	OBFORMAT
422	NOOBFORMAT
423	OUTLINE
424	NOOUTLINE
425	PARTITIONS
426	NOPARTITIONS
427	PRINTADAPT
428	NOPRINTADAPT
429	PROGSYMBOL
430	NOPROGSYMBOL
431	ALLQUERY
432	NOQUERY
433	COLDQUERY
434	SOSI
435	NOSOSI
436	TEXTKYBD
437	NOTEXTKYBD
438	TEXTPRINT
439	NOTEXTPRINT
440	VALIDATION

Value	CVDA
441	NOVALIDATION
442	RELREQ
443	NORELREQ
444	DISCREQ
445	NODISCREQ
446	ALTPRTCOPY
447	NOALTPRTCOPY
448	PRTCOPY
449	NOPRTCOPY
450	UCTRAN
451	NOUCTRAN
452	TRANIDONLY
460	MTOM
461	NOMTOM
462	SAMESENDMTOM
463	SENDMTOM
464	NOSENDMTOM
465	XOPDIRECT
466	NOXOPDIRECT
467	XOPSUPPORT
468	NOXOPSUPPORT
469	MTOMNOXOP
470	NOMTOMNOXOP
472	FEED
473	COLLECTION
474	CATEGORY
475	SERVICE
483	TEMPORARY
484	PERMANENT
485	NEWSESSION
486	OLDSESSION
487	NOSTSN
488	STSNSET
489	STSNTEST
490	EB
491	CD
492	MORE
493	LIC
494	RU
495	NOTCDEB
496	NONE

Value	CVDA
497	DEFRESP1
498	DEFRESP2
499	DEFRESP3
500	NOALARM
501	ALARM
502	FMH
503	NOFMH
504	PROTECTED
505	UNPROTECTED
506	MDT
507	NOMDT
508	DATA
509	STSN
510	BEGINSESSION
511	TIMEOUT
512	INSTALLFAIL
513	DISCARDFAIL
514	SETFAIL
515	ACQFAIL
516	SESSIONLOST
517	SESSIONFAIL
518	CONVIDLE
519	ADDFAIL
520	DELETEFAIL
521	UNSOLDATA
522	NORMALRESP
523	EXCEPTRESP
524	ATTENTION
525	LUSTAT
526	CANCEL
527	RTR
528	DEFRESP1OR2
529	POSITIVE
530	NEGATIVE
531	NOMSGJRNL
532	INOUT
533	T3278M2
534	T3278M3
535	T3278M4
536	T3278M5
537	T3279M2

Value	CVDA
538	T3279M3
539	T3279M4
540	T3279M5
541	LUP
542	FORMATTED
543	DATASTREAM
544	LOSE
545	WIN
546	NOTINBOUND
547	INBOUND
548	UNINSTALLED
549	RELEASING
550	INSTALLED
551	NOTINSTALLED
552	TPS55M2
553	TPS55M3
554	TPS55M4
555	TPS55M5
556	NOCONV
557	PENDSTSN
558	PENDBEGIN
559	APPLICATION
560	PENDDATA
561	PENDSTART
562	PENDRELEASE
563	RELEASE
564	PENDUNSOL
565	PENDPASS
566	SUMMUNLIKE
570	ASYNCHRONOUS
571	SYNCHRONOUS
572	NONTRANS
573	TRANS
574	CONTEXT
575	REGION
576	ATTACH
577	LINK
578	WMQ
579	CUSTOM
580	TRANSTART
581	CBER

Value	CVDA
582	CBE
583	CCE
584	CFE
585	WBE
600	CONVERSE
601	NOCONVERSE
602	SYNCPOINT
603	NOSYNCPOINT
604	GMT
605	LOCAL
606	FORMATEDF
607	NOFORMATEDF
608	NOTASKSTART
609	STARTED
610	STOPPED
611	TASKSTART
612	BUSY
613	NOTBUSY
614	SCS
615	DS3270
616	ASCII7
617	ASCII8
618	AUTOSTART
619	DRAINING
620	INDOUBT
621	INFLIGHT
622	WAITFORGET
623	CONTROLSHUT
624	CANCELLED
625	FIRSTINIT
626	SECONDINIT
627	THIRDINIT
628	INITCOMPLETE
629	STANDBY
630	AUTOACTIVE
631	AUTOINACTIVE
632	CTLGALL
633	CTLGMODIFY
634	CTLGNONE
635	START
636	POST

Value	CVDA
637	DELAY
638	ROUTE
640	CLEAR
641	NOCLEAR
642	USER
643	SYSTEM
644	SHUTENABLED
645	SHUTDISABLED
646	RFC1123
647	RFC3339
650	EXCI
651	GENERIC
652	SPECIFIC
655	FORCECANCEL
656	SYSPLEX
657	NOISOLATE
658	ISOLATE
659	NOTDEFINED
660	CICS
661	NONCICS
663	SUBSPACE
664	BASESPACE
665	XCF
666	REQUIRED
667	NOTREQUIRED
668	SOS
669	NOTSOS
670	REGISTERED
671	UNREGISTERED
672	UNAVAILABLE
673	CMDPROT
674	NOCMDPROT
675	RELATED
677	REGERROR
678	DEREGISTERED
679	DEREGERROR
680	REENTPROT
681	NOREENTPROT
682	SOSBELOW
683	SOSABOVE
684	DAE

Value	CVDA
685	NODAE
690	CONNECTED
691	NOTCONNECTED
693	SPI
694	NOSPI
696	PRECOMMAND
697	POSTCOMMAND
698	PROGRAMINIT
700	NORECOVDATA
701	RECOVDATA
702	RESYNC
703	UNCONNECTED
706	IMMQUIESCED
707	QUIESCED
708	QUIESCING
709	UNQUIESCED
710	NOLOSTLOCKS
711	NORETAINED
712	RECOVERLOCKS
713	REMLOSTLOCKS
714	RESETLOCKS
715	RETAINED
716	RETRY
720	RLS
721	NOTRLS
722	UNCOMMITTED
723	CONSISTENT
724	REPEATABLE
730	RLSACTIVE
731	RLSINACTIVE
732	VRRDS
734	NOCONVERT
735	CHUNKNO
736	CHUNKYES
737	CHUNKEND
738	SRVCONVERT
739	NOSRVCONVERT
740	REMSESSION
741	CLOSE
742	NOCLOSE
743	CLICONVERT

Value	CVDA
744	NOCLICONVERT
745	NOINCONVERT
746	NOOUTCONVERT
747	EXPECT
748	GET
749	PUT
750	HEAD
751	EXECENQ
752	EXECENQADDR
753	OWNER
754	WAITER
755	CONNECTION
756	DATASET
757	HEURCOMMIT
758	HEURBACKOUT
759	OPTIONS
760	TRACE
761	RLSSERVER
762	SHUNTED
763	WAITCOMMIT
765	WAITING
766	WAITRMI
767	TDQ (TDQUEUE)
768	TSQ (TSQUEUE)
770	COORDINATOR
771	RMI
772	STARTING
773	SUBORDINATE
774	NRS
774	UNKNOWN
775	NORMI
780	MVS
781	DUMMY
782	FAILED
783	FLUSH
784	NOSYSLOG
785	SYSLOG
786	KEEP
788	COLD
789	INITIAL
790	ENDAFFINITY

Value	CVDA
791	CACHE
792	COMMITFAIL
793	DATASETFULL
794	DEADLOCK
795	DELEXITERROR
796	INDEXRECFULL
797	IOERROR
798	OPENERROR
799	RLSGONE
800	BACKUPNONBWO
802	TCPIP
810	SKIP
811	LEAVE
812	REREAD
813	MOD
814	QUEUE
815	REJECT
816	IGNORERR
820	UNATTEMPTED
830	RRCOMMITFAIL
831	RRINDOUBT
832	LCKSTRUCFULL
833	CFTABLE
834	LOAD
835	NOLOAD
836	CONTENTION
837	LOCKING
840	CSDAPI
841	CSDBATCH
842	GRPLIST
843	DREPAPI
844	CREATESPI
845	DREPBATCH
847	TABLE
848	CLOUD
850	FROMEPR
851	TOEPR
852	REPLYTOEPR
853	FAULTTOEPR
854	REQCONTEXT
856	ALL

Value	CVDA
857	METADATA
858	REFPARMS
859	ADDRESS
860	ALLVALUES
861	DOESNOTEQUAL
862	DOESNOTSTART
863	EQUALS
864	GREATERTHAN
865	ISNOTGREATER
866	ISNOTLESS
867	LESSTHAN
868	STARTSWITH
869	CONTAINER
870	CURRENTPGM
871	EXISTS
872	DOESNOTEXIST
873	GOHIGHERTHAN
874	GOLOWERTHAN
900	ABEND
902	CGROUP
903	CONNECT
904	CONNECTING
905	CSIGN
906	CTERM
907	CTX
908	COPID
909	CUSERID
910	DISCONNING
911	EQUAL
912	EXIT
913	GROUP
914	HIGH
915	LOW
916	NOCONNECT
917	NOEXIT
918	NORELEASE
919	NOROLLBACK
920	NOTWAIT
921	N906D
922	POOL
923	REBUILD

Value	CVDA
924	RECONNECT
925	SIGN
926	SQLCODE
927	TWAIT
928	TXID
929	TX
930	USERID
931	N906
932	TPOOL
933	OPID
934	NORESYNC
935	BRIDGE
938	MRO
939	RRSUR
940	RZINSTOR
941	SCHEDULER
942	SOCKET
943	STARTTERM
944	TRANDATA
945	WEB
946	XMRUNTRAN
947	GROUPRESYNC
990	DEFINITION
991	EXPORT
992	IMPORT
993	ENTRYPOINT
994	POLICYSCOPE
997	NOPS
998	SNPS
999	MNPS
1000	NOTFIRED
1001	FIRED
1002	ACTIVITY
1003	COMPOSITE
1004	TIMER
1005	AND
1006	OR
1010	PROCESS
1011	NOUSER
1013	FORCED
1014	INCOMPLETE

Value	CVDA
1015	UNEXPECTED
1016	NORMAL
1017	EXPIRED
1018	UNEXPIRED
1019	CHAR
1020	BIT
1021	NOTDYNAMIC
1022	NOTROUTABLE
1023	ROUTABLE
1024	DORMANT
1025	CANCELLING
1026	COMPLETE
1027	NOTSUSPENDED
1029	HTTPS
1030	SSL
1031	NOSSL
1032	CLIENTAUTH
1033	HTTPYES
1034	HTTPNO
1035	PDSMEMBER
1036	APPEND
1037	NOAPPEND
1038	BINARY
1039	EBCDIC
1040	HFSFILE
1041	PROVIDER
1042	REQUESTER
1044	SOAP
1045	SENDER
1046	RECEIVER
1047	CHANNEL
1048	COMMAREA
1049	UPDATING
1050	QUASIRENT
1051	THREADSAFE
1052	BASEAPI
1052	CICSAPI
1053	OPENAPI
1054	NOFORCE
1055	CKOPEN
1056	UKOPEN

Value	CVDA
1057	QR
1058	INTERNAL
1060	SECUSER
1061	CERTUSER
1063	WAITRRMS
1066	NOTHOTPOOL
1068	XPLINK
1070	DEFAULTUSER
1071	NOAUTO
1072	PROGAUTO
1073	FULLAUTO
1074	IDENTIFY
1075	CLASSCACHE
1076	NOCLASSCACHE
1077	TERMINATE
1078	RELOAD
1079	RELOADING
1080	JVM
1081	NOJVM
1082	DEBUG
1083	NODEBUG
1084	NONLE370
1085	PUBLISH
1086	RETRACT
1087	CLIENTCERT
1088	SCAN
1089	IPIC
1090	NONCRITICAL
1091	NOAUTHENTIC
1092	BASICAUTH
1093	CERTIFICAUTH
1094	AUTOREGISTER
1095	AUTOAUTH
1096	HTTP
1097	IIOP
1098	ECI
1099	VERIFY
1100	BOTH
1101	CORBA
1102	EJB
1103	HOME

Value	CVDA
1104	ASSERTED
1105	KERBEROS
1106	SUPPORTED
1107	OTS
1109	RESPECTED
1110	NOPHASEOUT
1111	PHASEOUT
1112	INITING
1113	PENDINIT
1114	PENDRESOLVE
1115	RESOLVING
1116	UNRESOLVED
1117	UNUSABLE
1118	DISCARDING
1119	ATOM
1120	AUTOTERMID
1121	URMTERMID
1122	CLIENT
1123	SERVER
1124	PIPELINE
1125	DISABLEDHOST
1126	ANALYZER
1127	NOANALYZER
1128	EXTENDED
1129	NOTEXTENDED
1130	COMPRESS
1131	NOCOMPRESS
1140	AUTOINSTALL
1141	CORBASERVER
1142	DB2CONN
1143	DB2ENTRY
1144	DISPATCHER
1145	DOCTEMPLATE
1146	ENQUEUE
1147	JOURNALNAME
1148	JOURNALNUM
1149	JVMPOOL
1150	JVMPROFILE
1151	JVMPROGRAM
1152	LSRPOOL
1153	MONITOR

Value	CVDA
1154	MVSTCB
1155	NODE
1156	RECOVERY
1157	REQUESTMODEL
1158	STATS
1159	STORAGE
1160	STREAMNAME
1161	SUBPOOL
1162	SYSDUMPCODE
1163	TABLEMGR
1164	TASKSUBPOOL
1166	TCPIPSERVICE
1168	TEMPLATENAME
1169	TRANCLASS
1170	TRANDUMPCODE
1171	TRANSACTION
1173	URIMAP
1174	WEBSERVICE
1175	MQCONN
1176	IPCONN
1177	LIBRARY
1178	PROGRAMDEF
1179	ATOMSERVICE
1180	BUNDLE
1182	PARTNER
1183	TSMODEL
1184	DBTRAN
1185	JOURNALMODEL
1186	TYPETERM
1187	SESSIONS
1188	ENQMODEL
1189	DJAR
1190	PROCESSTYPE
1191	EVENTBINDING
1192	EVENTPROCESS
1193	JVMSERVER
1194	XMLTRANSFORM
1196	EPADAPTER
1197	MESSAGEID
1198	LOGREPLICATE
1199	NOLOGREPLICA

Value	CVDA
1200	EPADAPTERSET
1201	RESIDENT
1202	NONRESIDENT
1203	UTC
1204	SOMEAVAIL
1206	DSIE
1207	MQMONITOR
1208	NOAUTOSTART
1209	ASRUNTRAN
1210	RREPL
1211	REPLICATOR
1212	NOREPLICATOR
1214	SECERROR

Note: VTAM is now z/OS Communications Server.

Chapter 6. National language codes

Language codes are held as one character for NATLANG and NATLANGINUSE, and three characters for LANGUAGECODE and LANGINUSE.

Table 20. CICS language suffixes

Suffix	IBM Code	Language name
A	ENG	UK English
B	PTB	Brazilian Portuguese
C	CHS	Simplified Chinese
D	DAN	Danish
E	ENU	US English
F	FRA	French
G	DEU	German
H	KOR	Korean
I	ITA	Italian
J	ISL	Icelandic
K	JPN	Japanese
L	BGR	Bulgarian
M	MKD	Macedonian
N	NOR	Norwegian
O	ELL	Greek
P	PTG	Portuguese
Q	ARA	Arabic
R	RUS	Russian
S	ESP	Spanish
T	CHT	Traditional Chinese
U	UKR	Ukrainian
V	SVE	Swedish
W	FIN	Finnish
X	HEB	Hebrew
Y	SHC	Serbo-Croatian (Cyrillic)
Z	THA	Thai
1	BEL	Byelorussian
2	CSY	Czech
3	HRV	Croatian
4	HUN	Hungarian
5	PLK	Polish
6	ROM	Romanian
7	SHL	Serbo-Croatian (Latin)
8	TRK	Turkish

© Copyright IBM Corp. 1989, 2017

Table 20. CICS language suffixes (continued)

Suffix	IBM Code	Language name
9	NLD	Dutch

There are other IBM codes not supported by CICS.

Table 21. Other IBM language codes

IBM Code	Language name
AFR	Afrikaans
CAT	Catalan
DES	Swiss German
ENA	Australian English
ENP	English Upper Case
FRB	Belgian French
FRC	Canadian French
FRS	Swiss French
GAE	Irish Gaelic
ITS	Swiss Italian
NLB	Belgian Dutch - Flemish
NON	Norwegian - Nynorsk
RMS	Rhaeto-Romanic
SKY	Slovakian
SLO	Slovenian
SRL	Serbian (Latin)
SRB	Serbian (Cyrillic)
SQI	Albanian
URD	Urdu

Chapter 7. Terminal control

This reference information applies to all terminals and logical units. For more detail, see the command descriptions.

Commands and options for terminals and logical units

This section describes the commands and options that apply to terminals and logical units.

Fullword lengths

For all terminal control commands, fullword length options can be used instead of halfword length options. In particular, where the following options are used in CONVERSE, RECEIVE, or SEND, the corresponding alternative can be specified instead (except for those noted):

Option	Alternative
LENGTH	FLENGTH
TOLENGTH	TOFLENGTH
FROMLENGTH	FROMFLENGTH
MAXLENGTH	MAXFLENGTH

Application programs must be consistent in their use of fullword and halfword options on terminal control commands. The maximum value that can be specified as a parameter on any length keyword is 32 767.

Read from terminal or logical unit (RECEIVE)

The RECEIVE command is used to read data from a terminal or logical unit. The INTO option is used to specify the area into which the data is to be placed. Alternatively, a pointer reference can be specified in the SET option. CICS acquires an area large enough to hold the data and sets the pointer reference to the address of that data.

The contents of this area are available to the task until the next terminal I/O command. However, the area does not belong to the task and is released by CICS while processing the next request. Therefore, this area cannot be passed back to CICS for further processing.

The application can use MAXLENGTH to specify the maximum length of data that the program accepts. If the MAXLENGTH option is omitted on a RECEIVE command for which the INTO option is specified, the maximum length of data the program accepts can be specified in the LENGTH option. If the MAXLENGTH option is omitted on a RECEIVE command for which the SET option is specified, CICS acquires enough storage to hold all the available data.

If the data exceeds the specified maximum length and the NOTRUNCATE option is specified, the remaining data is made available to satisfy subsequent RECEIVE commands. If NOTRUNCATE is not specified, the data is truncated and the LENGERR condition occurs. In this event, if the LENGTH option is specified, the

© Copyright IBM Corp. 1989, 2017

named data area is set to the actual data length (before truncation occurs) when data has been received. The first RECEIVE command in a task started by a terminal does not issue a terminal control read but copies the input buffer, even if the data length is zero. A second RECEIVE command must be issued to cause a terminal control read.

When a PA key is defined as a print key by the system initialization parameter PRINT, and that key is pressed in response to a RECEIVE command, it has no effect on the application program. The RECEIVE command is satisfied, and the application allowed to continue, when another attention (that is, one of the other PA keys, any of the PF keys, the ENTER key, or the light pen) is made at the keyboard.

Write to terminal or logical unit (SEND)

The SEND command is used to write data to a terminal or logical unit. The options FROM and LENGTH specify the data area from which the data is to be taken and the length (in bytes) of the data. For a transaction started by automatic transaction initiation (ATI), a SEND command should always precede the first RECEIVE in a transaction.

Unless the WAIT option of the SEND command is specified also, the transmission of the data associated with the SEND command is deferred until a later event, such as a sync point, occurs. This deferred transmission reduces the flows of data by allowing data flow controls to be transmitted with the data.

Transmission is not deferred for distributed transaction processing when interregion communication (IRC) is in use.

Synchronize terminal I/O for a transaction (WAIT TERMINAL)

This command is used to ensure that a terminal operation has completed before further processing occurs in a task under which more than one terminal or logical unit operation is performed. Alternatively, the WAIT option can be specified in a SEND command. (A wait is always carried out for a RECEIVE command.) Either method may cause execution of a task to be suspended. If suspension is necessary, control is returned to CICS. Execution of the task is resumed when the operation is completed.

Even if the WAIT option is not specified in a SEND command, the EXEC interface program ensures that the operation is completed before issuing a subsequent RECEIVE or SEND command.

Converse with terminal or logical unit (CONVERSE)

For most terminals or logical unit types, a conversational mode of communication can be used. The CONVERSE command is used for this purpose and means that the 3650 application program communicates with the host processor. If this option is not specified, the 3650 application program cannot communicate with the host processor. In general, the CONVERSE command can be considered as a combination of a SEND command followed immediately by a WAIT TERMINAL command and then by a RECEIVE command. However, not all options of the SEND and RECEIVE commands are valid for the CONVERSE command; specific rules are given in the syntax descriptions for different devices. The TOLENGTH option is equivalent to the LENGTH option of the RECEIVE command, and the

FROMLENGTH option is equivalent to the LENGTH option of the SEND command.

Send an asynchronous interrupt (ISSUE SIGNAL)

This command is used, in a transaction in receive mode, to signal to the sending transaction that a mode change is needed. The execution of the command raises the SIGNAL condition on the next SEND or RECEIVE command executed in the sending transaction, and a previously executed HANDLE CONDITION command for this condition can be used either to action the request or to ignore it.

Disconnect a switched line (ISSUE DISCONNECT)

This command is used to break a line connection between a terminal and the processor, or to break a session between SNA logical units (LUs), when the transaction is completed. If the terminal is a buffered device, the data in the buffers is lost.

When used with an SNA LU, ISSUE DISCONNECT, which does not become effective until the task completes, signs off the terminal, frees the COMMAREA, clears the next TRANID, stops any BMS paging, and, if autoinstall is in effect, deletes the terminal definition.

Teletypewriter programming

The Teletypewriter (World Trade only) uses two different control characters for print formatting.

These control characters are as follows:

```
<   carriage return (X'22' in ITA2
    code or X'15' in EBCDIC)

=   line feed (X'28' in ITA2 code
     or X'25' in EBCDIC)
```

Use the character < first; otherwise following characters (data) can be printed while the type bar is moving to the left.

Message format

To begin a new message on a new line at the left margin, start the message text with X'1517' (EBCDIC). CICS recognizes the X'17' and changes it to X'25' (X'17' is an IDLE character).

In the message body, to write several lines with a single transmission, the lines must be separated by X'1525', or if multiple blank lines are required, by X'152525...25'.

To allow input of the next message on a new line at the left margin, the preceding message must end with X'1517'. CICS recognizes X'15' and changes the character following it to X'25'.

If two or more successive output messages have “message begin” and “message end” which look the same, to make the “message end” of the preceding message

distinguishable from the “message begin” of the next message, the next to last character of the “message end” must not be X'15'.

Message length

For messages for teletypewriter terminals do not exceed a length of about 3000 bytes or approximately 300 words.

Connection through z/OS Communications Server

Both the TWX Model 33/35 Common Carrier Teletypewriter Exchange and the WTTY Teletypewriter (World Trade only) can be connected to CICS through z/OS Communications Server using NTO.

If a device is connected through z/OS Communications Server using NTO, the protocols used are the same as for the 3767 logical unit, and the application program can use these protocols (for example, HANDLE CONDITION SIGNAL). However, the data stream is not translated to a 3767 data stream but remains as that for a TWX/WTTY.

Display device operations

In addition to the standard terminal control commands for sending and receiving data, several commands and lists are provided for use with display devices such as the 3270.

The commands are:

- Print displayed information (ISSUE PRINT)
- Copy displayed information (ISSUE COPY)
- Erase all unprotected fields (ISSUE ERASEAUP)
- Handle input without data (RECEIVE)
- Handle attention identifiers (HANDLE AID)

The lists are:

- Standard attention identifier list (DFHAID)
- Standard attribute and printer control character list (DFHBMSCA)

For devices with switchable screen sizes, the size of the screen that can be used, and the size to be used for a given transaction, are defined by CICS table generation. These values can be obtained with the ASSIGN command, described in “ASSIGN” on page 41.

Always include the ERASE option in the first SEND command, to clear the screen and format it according to the transmitted data. This first SEND with ERASE also selects the screen size to be used, as specified using the RDO option SCRNSIZE, or in the TCT. If ERASE is omitted, the screen size is the same as its previous setting, which can be incorrect.

Use the CLEAR key outside of a transaction to set the screen to its default size.

Print displayed information (ISSUE PRINT)

ISSUE PRINT prints displayed data on the first available printer that is eligible to respond to a print request.

For a 3270 logical unit or a 3650 host-conversational (3270) logical unit, it is a printer defined by the RDO TERMINAL options PRINTER and ALTPRINTER, or by a printer supplied by the autoinstall user program.

For a 3270-display logical unit with the printer adapter feature (PRINTADAPTER(YES) option on RDO TYPETERM), used with a 3274 or 3276, it is a printer allocated by the printer authorization matrix. See *An Introduction to the IBM 3270 Information Display System*.

For a 3790 (3270-display) logical unit, it is a printer allocated by the 3790.

For a printer to be available, it must be in service and not currently attached to a task.

For a 3270 logical unit to be eligible, it must have been specified by RDO TERMINAL options PRINTER and ALTPRINTER or by a printer supplied by the autoinstall user program, and it must have the correct buffer capacity. If the copy feature is also specified (COPY(YES) on RDO TYPETERM definition the printer must be on the same control unit.

If an ISSUE PRINT command is executed, the printer involved must be owned by the same CICS system that owns the terminal that is running the transaction.

For some 3270 displays, it is possible also to print the displayed information without using CICS. See *An Introduction to the IBM 3270 Information Display System* manual.

Copy displayed information (ISSUE COPY)

The ISSUE COPY command is used to copy the format and data contained in the buffer of a specified terminal into the buffer of the terminal that started the transaction.

This command cannot be used for an LUTYPE2 connection. Both terminals must be attached to the same remote control unit. The terminal whose buffer is to be copied is identified in the TERMID option. If the terminal identifier is not valid, that is, it does not exist in the TCT, then the TERMIDERR condition occurs. The copy function to be performed is defined by the copy control character (CCC) specified in the CTLCHAR option of the ISSUE COPY command.

The WAIT option of the ISSUE COPY command ensures that the operation has been completed before control is returned to the application program.

Erase all unprotected fields (ISSUE ERASEAUP)

The ISSUE ERASEAUP command is used to erase all unprotected fields of a 3270 buffer.

It achieves this using the following actions:

1. All unprotected fields are cleared to nulls (X'00').
2. The modified data tags (MDTs) in each unprotected field are reset to zero.
3. The cursor is positioned to the first unprotected field.
4. The keyboard is restored.

The WAIT option of the ISSUE ERASEAUP command ensures that the operation has been completed before control is returned to the application program.

Handle input without data (RECEIVE)

The RECEIVE command with no options causes input to take place and the EIB to be updated.

Data received by CICS is not passed on to the application program and is lost. A wait is implied. Two of the fields in the EIB that are updated are EIBCPOSN and EIBAID.

Cursor position (EIBCPOSN)

For every terminal control (or BMS) input operation associated with a display device, the screen cursor address (position) is placed in the EIBCPOSN field in the EIB. The cursor address is in the form of a halfword binary value and remains until updated by a new input operation.

Attention identifier (EIBAID)

For every terminal control (or BMS) input operation associated with a display device, an attention identifier (AID) is placed in field EIBAID in the EIB.

The AID indicates which method the terminal operator has used to initiate the transfer of information from the device to CICS; for example, the ENTER key, a program function key, the light pen, and so on. The field contents remain unaltered until updated by a new input operation.

Field EIBAID can be tested after each terminal control (or BMS) input operation to determine further processing, and a standard attention identifier list (DFHAID) is provided for this purpose. Alternatively, the HANDLE AID command can be used to pass control to specified labels when the AIDs are received.

EIBAID and EIBCPOSN are also updated at task initiation for non-ATI tasks and after each terminal control and BMS input.

Chapter 8. Common Programming Interface Communications (CPI Communications)

Common Programming Interface Communications (CPI Communications) is the communication element of the Systems Applications Architecture (SAA) Common Programming Interface (CPI).

CPI Communications in CICS provides an alternative application programming interface (API) to existing CICS Advanced Program-to-Program Communications (APPC) support. CPI Communications provides distributed transaction processing (DTP) on APPC sessions and can be used in assembler language, COBOL, PL/I, or C.

CPI Communications defines an API that can be used in APPC networks that include multiple system platforms, where the consistency of a common API is of benefit.

The CPI Communications interface can converse with applications on any system that provides an APPC API. This includes applications on CICS platforms. You can use APPC API commands on one end of a conversation and CPI Communications commands on the other. CPI Communications requires specific information (side information) to begin a conversation with a partner program. CICS implementation of side information is achieved using the partner resource, which your system programmer is responsible for maintaining.

Calls from the application to the CPI Communications interface are resolved by link-editing it with the CICS CPI Communications stub (DFHCPLC). For information abut how to do this, see Including the CICS-supplied interface modules in Deploying.

The CPI Communications API is defined as a general call interface. The interface is described in the *Common Programming Interface Communications Reference*.

SAA Resource Recovery

SAA Resource Recovery is the recovery element of the Systems Application Architecture® (SAA) Common Programming Interface (CPI).

SAA Resource Recovery provides that architecture's alternative application programming interface (API) to EXEC CICS SYNCPOINT and EXEC CICS SYNCPOINT ROLLBACK functions in CICS. (See the *SAA Common Programming Interface-Resource Recovery Reference*, SC31-6821, for more details.)

CICS supports only those SAA Resource Recovery return codes that match existing EXEC CICS commands. This leaves only two return codes: RR_OK and RR_BACKED_OUT.

SRRCMT

The SAA Resource Recovery commit call SRRCMT (equivalent to EXEC CICS SYNCPOINT) has the following return codes:

- RR_OK

© Copyright IBM Corp. 1989, 2017

- RR_COMMITTED_OUTCOME_PENDING
- RR_COMMITTED_OUTCOME_MIXED
- RR_PROGRAM_STATE_CHECK
- RR_BACKED_OUT
- RR_BACKED_OUT_OUTCOME_PENDING
- RR_BACKED_OUT_OUTCOME MIXED

Because of the restriction, these are replaced by:
- RR_COMMITTED_OUTCOME_PENDING, RR_OK
- RR_COMMITTED_OUTCOME_MIXED, RR_OK
- RR_PROGRAM_STATE_CHECK, shown as abend code ASP2
- RR_BACKED_OUT_OUTCOME_PENDING, RR_BACKED_OUT
- RR_BACKED_OUT_OUTCOME MIXED, RR_BACKED_OUT

SRRBACK

The SAA Resource Recovery backout call SRRBACK (equivalent to EXEC CICS SYNCPOINT ROLLBACK) has the following return codes:
- RR_OK
- RR_COMMITTED_OUTCOME_PENDING
- RR_COMMITTED_OUTCOME_MIXED

Because of the restriction, all these are replaced by RR_OK.

Chapter 9. BMS-related constants

This information describes the BMS-related standard attribute and printer control characters, a bitmap for attributes, MSR control value constants, and attention identifier constants.

The standard list DFHBMSCA makes it simpler to provide field attributes and printer control characters. Table 22 lists the symbolic names for the various combinations of attributes and control characters. If you need combinations other than the ones shown, you must generate them separately. To help you do this, see Table 23 on page 851 for a bitmap of attributes. To find the value of an attribute constant, see the *3274 Control Unit Reference Summary*.

You can get the standard attribute and printer character control list by copying copybook DFHBMSCA into your application.

- For COBOL users, it consists of a set of 01 statements that can be copied into the working storage section.
- For C users, it is included in applications as follows:

```
#include "dfhbmsca.h"
```

- For PL/I users, it consists of DECLARE statements defining elementary character variables.
- For Assembler language users, the list consists of a set of EQU statements.

You must use the symbolic name DFHDFT in the application structure to override a map attribute with the default. You can use a high value, such as X'FF', to reset the COLOR, HILIGHT, OUTLINE, PS, SOSI, or VALIDN attributes to their default values. To specify default values in a set attribute (SA) sequence in text build, use the symbolic names DFHDFCOL, DFHBASE, or DFHDFHI.

Table 22. Standard attribute and printer control character list, DFHBMSCA

Constant	Meaning
DFHBMPEM	Printer end-of-message
DFHBMPNL	Printer new-line
DFHBMPFF	Printer form feed
DFHBMPCR	Printer carriage return
DFHBMASK	Autoskip
DFHBMUNP	Unprotected
DFHBMUNN	Unprotected and numeric
DFHBMPRO	Protected
DFHBMBRY	Bright
DFHBMDAR	Dark
DFHBMFSE	MDT set
DFHBMPRF	Protected and MDT set
DFHBMASF	Autoskip and MDT set
DFHBMASB	Autoskip and bright
DFHBMPSO	shift-out value X'0E'.
DFHBMPSI	shift-in value X'0F'.

© Copyright IBM Corp. 1989, 2017

Table 22. Standard attribute and printer control character list, DFHBMSCA (continued)

Constant	Meaning
DFHBMEOF	Field erased
DFHBMCUR	Field containing cursor flagged
DFHBMEC	Erased field containing cursor (COBOL only)
DFHBMFLG	Flags (COBOL only)
DFHBMDET	Field detected
DFHSA[1]	Set attribute (SA) order
DFHERROR	Error code
DFHCOLOR[1]	Color
DFHPS[1]	Programmed symbols
DFHHLT[1]	Highlight
DFH3270[1]	Base 3270 field attribute
DFHVAL	Validation
DFHOUTLN	Field outlining attribute code
DFHBKTRN	Background transparency attribute code
DFHALL[1]	Reset all to defaults
DFHDFT	Default
DFHDFCOL[1]	Default color
DFHBLUE	Blue
DFHRED	Red
DFHPINK	Pink
DFHGREEN	Green
DFHTURQ	Turquoise
DFHYELLO	Yellow
DFHNEUTR	Neutral
DFHBASE[1]	Base programmed symbols
DFHDFHI[1]	Normal
DFHBLINK	Blink
DFHREVRS	Reverse video
DFHUNDLN	Underscore
DFHMFIL[2]	Mandatory fill
DFHMENT[2]	Mandatory enter
DFHMFE	Mandatory fill and mandatory enter
DFHMT	Trigger
DFHMFT	Mandatory fill and trigger
DFHMET	Mandatory enter and trigger
DFHMFET	Mandatory fill and mandatory enter and trigger
DFHUNNOD	Unprotected, nondisplay, nonprint, nondetectable, MDT
DFHUNIMD	Unprotected, intensify, light-pen detectable, MDT
DFHUNNUM	Unprotected, numeric, MDT
DFHUNNUB	Unprotected, numeric, intensify, intensify, light-pen detectable
DFHUNINT	Unprotected, numeric, intensify, light-pen detectable, MDT
DFHUNNON	Unprotected, numeric, nondisplay, nonprint, nondetectable, MDT
DFHPROTI	Protected, intensify, light-pen detectable
DFHPROTN	Protected, nondisplay, nonprint, nondetectable
DFHDFFR	Default outline
DFHUNDER	Underline
DFHRIGHT	Right vertical line
DFHOVER	Overline
DFHLEFT	Left vertical line
DFHBOX	Underline and right vertical and overline and left vertical
DFHSOSI	SOSI=yes
DFHTRANS	Background transparency

Table 22. Standard attribute and printer control character list, DFHBMSCA (continued)

Constant	Meaning
DFHOPAQ	No background transparency

Notes:

[1] For text processing only. Use for constructing embedded set attribute orders in user text.

[2] Cannot be used in set attribute orders.

Table 23. Bitmap for attributes

prot	a/n	hi	spd	ndp	mdt	ebcd	asci	char
U						40	20	b (blank)
U					Y	C1	41	A
U			Y			C4	44	D
U			Y		Y	C5	45	E
U		H	Y			C8	48	H
U		H	Y		Y	C9	49	I
U				Y		4C	3C	<
U				Y	Y	4D	28	(
U	N					50	26	
U	N				Y	D1	4A	J
U	N		Y			D4	4D	M
U	N		Y		Y	D5	4E	N
U	N	H	Y			D8	51	Q
U	N	H	Y		Y	D9	52	R
U	N			Y		5C	2A	*
U	N			Y	Y	5D	29	)
P						60	2D	- (hyphen)
P					Y	61	2F	/
P			Y			E4	55	U
P			Y		Y	E5	56	V
P		H	Y			E8	59	Y
P		H	Y		Y	E9	5A	Z
P				Y		6C	25	%
P				Y	Y	6D	5F	_ (underscore)
P	S					F0	30	0
P	S				Y	F1	31	1
P	S		Y			F4	34	4
P	S		Y		Y	F5	35	5
P	S	H	Y			F8	38	8
P	S	H	Y		Y	F9	39	9
P	S			Y		7C	40	@
P	S			Y	Y	7D	27	'

Table 24. Key to attributes and settings in bitmap

Code	Meaning
a/n	Automatic skip or numeric

Table 24. Key to attributes and settings in bitmap (continued)

Code	Meaning
asci	American National Standard Code for Information Interchange
char	Graphic character equivalent to hex code
ebcd	Extended binary coded decimal interchange code
hi	High intensity
H	High
mdt	modified data tag
ndp	nondisplay print
N	Numeric
prot	Protected
P	Protected
spd	Selector pen detectable
S	Automatic skip
U	Unprotected
Y	Yes

Magnetic slot reader (MSR) control value constants, DFHMSRCA

A selection of MSR control value constants has been created for CICS and stored in copybook DFHMSRCA. The patterns are stored as named constants that can be loaded by simple application program commands. Provision of such constants saves the programmer from having to build a commonly used bit pattern whenever it is required.

MSR control byte values

A selection of MSR control byte values has been created for CICS and stored in the copybook DFHMSRCA. The following table shows you the meaning of each bit.

The constants supplied in DFHMSRCA are listed in Table 25.

Table 25. Standard list DFHMSRCA

Constant	Meaning
DFHMSRST	MSR reset. All lights and buzzers off. MSR available for input.
DFHMSCON	Transaction ready for more input. Green and yellow on; emit short buzz; IN PROCESS (user) mode set.
DFHMSFIN	Input complete. Green on; emit short buzz; IN PROCESS mode reset.
DFHMSALR	Operator alert. Green, yellow, and red on; emit long buzz; IN PROCESS mode reset.
DFHMSALS	Operator alert. Green, yellow, and red on; emit long buzz; IN PROCESS mode set.
DFHMSIPY	IN PROCESS state set. Yellow on.
DFHMSIPN	IN PROCESS state reset.
DFHMSLKY	MSR operation inhibited. Yellow on.
DFHMSLKN	MSR input allowed. Green on. Yellow on.
DFHMSAEY	MSR autoenter on. Yellow on.
DFHMSAEN	MSR autoenter off. Yellow on.

Table 25. Standard list DFHMSRCA (continued)

Constant	Meaning
DFHMSLBN	Long buzzer suppressed. Yellow on.
DFHMSLBY	Long buzzer permitted. Yellow on.
DFHMSSBN	Short buzzer suppressed. Yellow on.
DFHMSSBY	Short buzzer permitted. Yellow on.
DFHMSNOP	Leave all MSR settings unchanged.

STATE MASK

If a bit is on in the STATE MASK byte, the state it represents is adopted by the device if the corresponding bit is also on in the STATE VALUE byte.

0 USER
User mode. Turn on the yellow light if the same bit is on in STATE VALUE.

1 LOCK
Locked/Unlocked. If locked, MSR input is inhibited.

2 AUTO
Autoenter on/off. If set on, any card read by the MSR causes an ENTER operation. If off, only a secure card causes an ENTER.

3 Ai1S
Suppress audible alarm 1.

4 Ai2S
Suppress audible alarm 2.

STATE VALUE

Modifies state to on or off if the corresponding bit is set on in STATE MASK.

INDICATOR MASK

Performs a similar function to STATE MASK, but for indicators.

0 Light 1 (Green)

1 Light 1 (Green)

2 Light2 (Yellow)

3 Audible alarm 1 (Long buzz)

4 Audible alarm 2 (Short buzz)

INDICATOR VALUE

Performs a similar function to STATE VALUE.

Attention identifier constants, DFHAID

The standard attention identifier list, DFHAID, simplifies testing the contents of the EIBAID field. The following table shows you the symbolic name for the attention identifier (AID) and the corresponding 3270 function.

You can get a copy of the list by copying DFHAID into your application program. For COBOL users, it consists of a set of 01 statements that must be copied into the working-storage section. For C users, it consists of a series of defined constants. For PL/I users, it consists of DECLARE statements defining elementary character variables.

Here is the table showing the names for the AID.

Table 1. Standard list DFHAID

Constant	Meaning
DFHENTER	ENTER key.
DFHCLEAR	CLEAR key.
DFHPA1–DFHPA3	PA1–PA3 keys.
DFHPF1–DFHPF24	PF1–PF24 keys.
DFHOPID	OPERID or MSR.
DFHMSRE	Extended (standard) MSR.
DFHTRIG	Trigger field.
DFHPEN	SELECTOR PEN or CURSOR SELECT key.
DFHCLRP[1]	CLEAR PARTITION key.
DFHSTRF[1]	Structured field pseudo-AID.

Note:

1. DFHCLRP and DFHSTRF do not apply to minimum function BMS.

Chapter 10. BMS macros

The syntax of each BMS macro is defined, separating the various operands and options into those appropriate to minimum, standard, and full BMS.

When coding, have the title in column 1, the macro in column 10, continuation lines should have * in column 72 and continue on column 16 on the next line.

For more information about BMS, see Basic mapping support.

Map set, map, and field definition

Ensure that the names of maps, and names of fields within a map set (or within multiple map sets that are copied into one application program) are unique. However, a map can have the same name as a map set.

Before CICS can load a physical map, it requires an installed resource definition for the map object. You can either use program autoinstall to create the definition when the map set is first used, or define a map set in the CSD using the DEFINE MAPSET resource definition.

You assemble a BMS map set definition to generate either a symbolic description map or a physical map. The physical map is a structured data area used at execution time to build the data stream for the terminal. The symbolic map is a series of data structures which you copy into your program at compile time so you can refer to the fields in the map by name.

For programming information about the autoinstall user program, see Writing a program to control autoinstall of LUs.

DFHMSD

The DFHMSD macro defines a map set.

DFHMDI

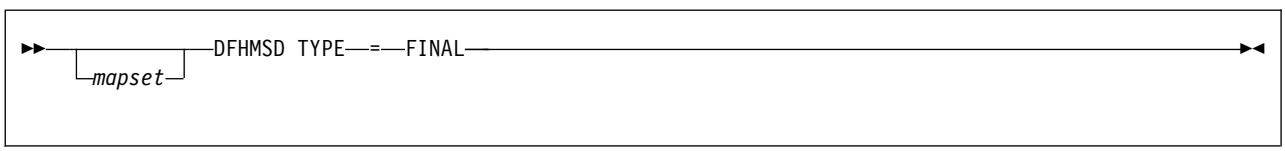
The DFHMDI macro defines a map within the map set defined by the previous DFHMSD macro. A map contains zero or more fields.

DFHMDF

The DFHMDF macro defines a field within a map defined by the previous DFHMDI macro.

Ending a mapset definition

A mapset definition ends with a macro of the form:



"mapset" is optional, but if used it must be the same as that on the DFHMSD macro that began the mapset definition.

© Copyright IBM Corp. 1989, 2017

ADS descriptor

Physical maps produced by CICS Transaction Server for z/OS also include an ADS descriptor in the output load module. This is provided to allow interpretation of the BMS Application Data Structure (the structure used by the application program for the data in SEND and RECEIVE MAP requests), without requiring your program to include the relevant DSECT or copybook at compile time.

The ADS descriptor contains a header with general information about the map, and a field descriptor for every field that appears in the ADS (corresponding to every named field in the map definition macro).

The ADS descriptor is generated for all maps. You can use the DSECT option to select the long form of the ADS, where all fields are aligned on 4 byte boundaries. The long form of the ADS is required by the 3270 bridge when an interface to WebSphere® MQ is used.

Partition set definition

Partitions are defined by coding the macros DFHPSD (partition set definition) and DFHPDI (partition definition). Each partition definition must be part of a partition set definition.

DFHPSD

Each partition set definition contains a single DFHPSD macro followed by one or more DFHPDI macros, and ending with a partition set definition TYPE=FINAL.

Before CICS can load a physical map, you must define a physical map using an RDO transaction with the MAPSET attribute.

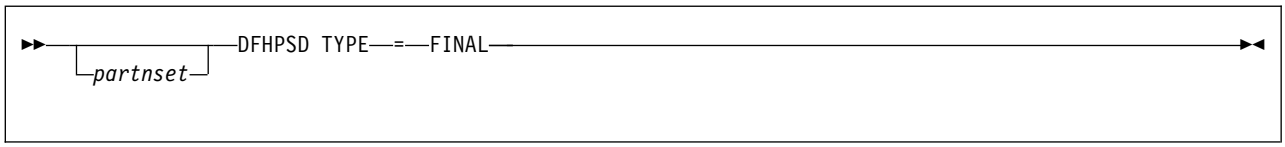
An alternative to defining maps using RDO is to use the program autoinstall exit to create the definition when the map set is first used. (For programming information about the autoinstall user program, see Writing a program to control autoinstall of LUs.)

DFHPDI

A partition set contains one or more partitions. Each partition is defined by coding a partition definition macro.

Ending a partition set definition

A partition set definition ends with a macro of the form:



The PARTNSET name (if specified) must match that specified on the DFHPSD macro that started the partition set definition.

Field groups

Very often, an output data display field has to contain several subfields, all sharing the same display attributes, and each of which might have to be modified separately.

At output, subfields that have not been modified by the program can adopt default data values from the output map. For example, a display can include a date field of a “day” subfield, “month” subfield, and “year” subfield. The contents of the year subfield remain constant over a relatively long period; its value can safely be taken from a map. However, the day value and month value must be updated regularly. Similarly, on input the terminal operator can enter data in each subfield separately.

You use the GRPNAME operand to define a group of subfields that combine to produce a field. The start of the group is indicated by a DFHMDF macro with the GRPNAME operand. This operand defines the first subfield, and specifies the attributes and name of the group. It is followed by other DFHMDF macros, one for each of the other subfields. Each of these must specify the group name, but cannot specify attribute values. The definition of the group is terminated by a DFHMDF macro that specifies a different group name, by one that specifies no group name, or by a DFHMDI or DFHMSD macro.

Briefly, a group of fields in a map would appear as follows in the map definition:

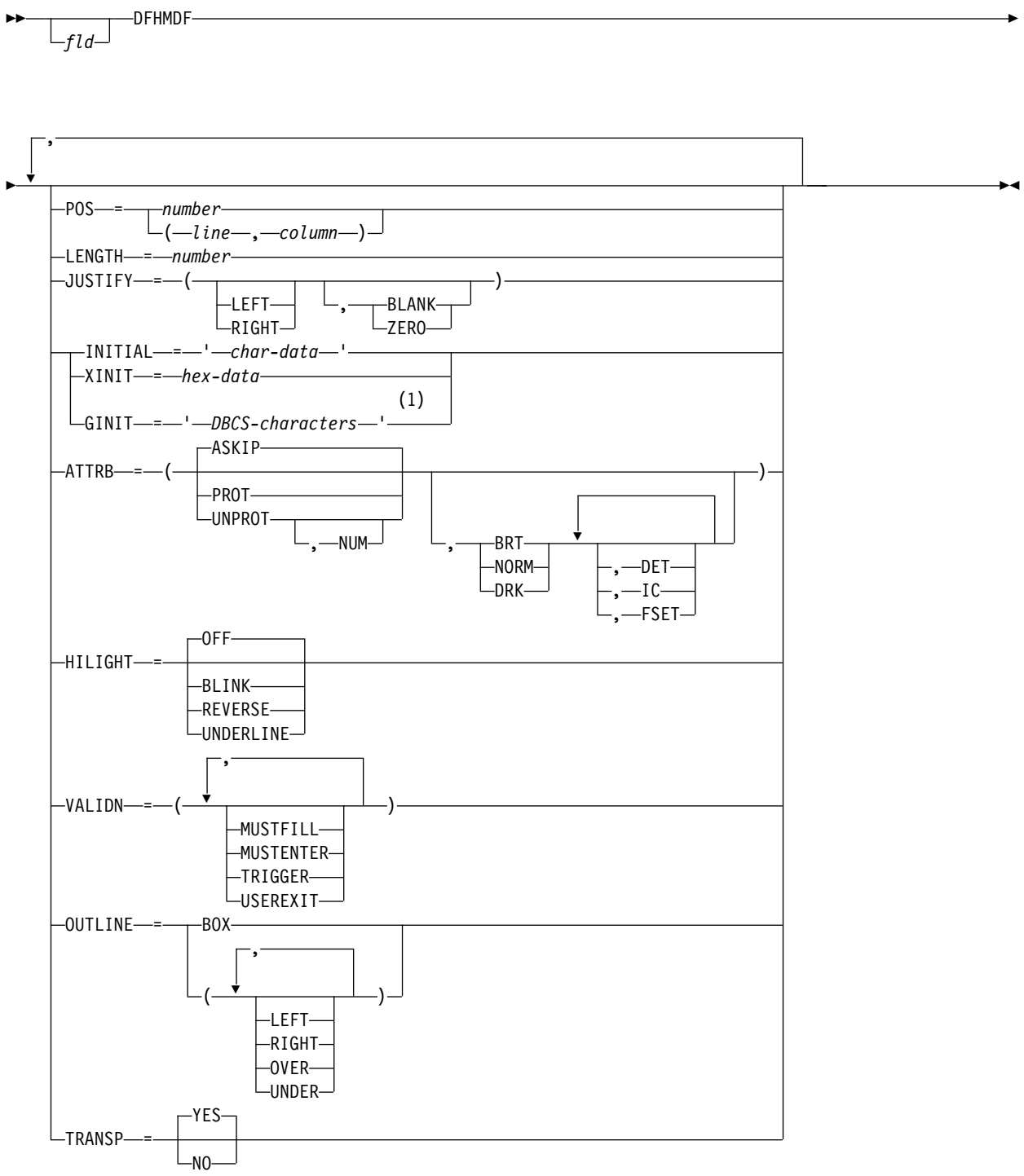
```
MAPSET DFHMSD....
       .
       .
MAP    DFHMDI....
       .
       .
DD     DFHMDF GRPNAME=DATE,POS=40,
              LENGTH=2,ATTRB=...
       .
MM     DFHMDF GRPNAME=DATE,POS=46,
              LENGTH=2
       .
YY     DFHMDF GRPNAME=DATE,POS=52,
              LENGTH=2
       .
FIELD  DFHMDF LENGTH=5,COLOR=GREEN,...
       DFHMSD TYPE=FINAL
```

The POS operand specifies the position of the attribute byte of the field even though subfields of a group, other than the first, do not have attributes. If the subfields are positioned contiguously with no intervening blanks, the POS of the second and succeeding subfields must specify the position of the last character of the previous subfield.

DFHMDF

The DFHMDF macro defines a field within a map defined by the previous DFHMDI macro.

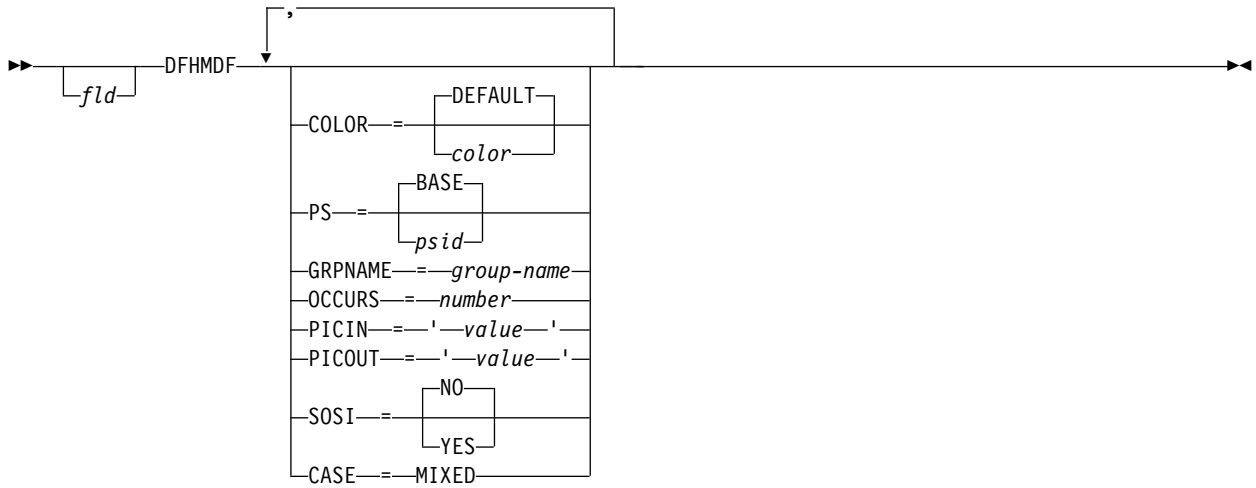
DFHMDF (part 1)



Notes:

1 DBCS characters start with a shift-out character X'0E' and end with a shift-in character X'0F'.

DFHMDF (part 2)



Description

A map contains zero or more fields.

fld is the name (1–30 characters) of the field. Refer to your compiler manual to ensure that there are no other restrictions on the length.

For more information about defining field names, see . If **fld** is omitted, application programs cannot access the field to change its attributes or alter its contents. For an output map, omitting the field name might be appropriate when the INITIAL operand is used to specify the contents of a field. If a field name is specified and the map that includes the field is used in a mapping operation, data supplied by the user overlays data supplied by initialization (unless default data only is being written).

The performance of input mapping operations is optimized if DFHMDF macros are arranged in numeric order of the POS operand.

You cannot define more than 1023 named fields for a COBOL, C, or PL/I input/output map.

You must ensure that the names of maps, and names of fields within a mapset (or within multiple mapsets that are copied into one application program) are unique. However, a map can have the same name as a mapset.

Before CICS can load a physical map, you must define a physical map using an RDO transaction with the MAPSET attribute.

Operands

ATTRB

This operand applies only to 3270 data stream devices; it is ignored for other devices, except that ATTRB=DRK is honored for the SCS Printer Logical Unit. It is also ignored (except for ATTRB=DRK) if the NLEOM option is specified on the SEND MAP command for transmission to a 3270 printer. In particular,

ATTRB=DRK must not be used as a method of protecting secure data on output on non-3270, non-SCS printer terminals.

If ATTRB is specified within a group of fields, it must be specified in the first field entry. A group of fields appears as one field to the 3270. Therefore, the ATTRB specification refers to all the fields in a group as one field rather than as individual fields. It specifies device-dependent characteristics and attributes, such as the capability of a field to receive data, or the intensity to be used when the field is output. It could however, be used for making an input field nondisplay for secure entry of a password from a screen. For input map fields, DET and NUM are the only valid options; all others are ignored.

ASKIP
is the default and specifies that data cannot be keyed into the field and causes the cursor to skip over the field.

BRT specifies that a high-intensity display of the field is required. Because of the 3270 attribute character bit assignments, a field specified as BRT is also potentially detectable. However, for the field to be recognized as detectable by BMS, DET must also be specified.

DET specifies that the field is potentially detectable.

The first character of a 3270 detectable field must be one of the following:

```
? > & blank
```

If ? or >, the field is a selection field; if & or blank, the field is an attention field. (See *An Introduction to the IBM 3270 Information Display System* for further details about detectable fields.)

A field for which BRT is specified is potentially detectable to the 3270, because of the 3270 attribute character bit assignments, but is not recognized as such by BMS unless DET is also specified.

DET and DRK are mutually exclusive.

If DET is specified for a field on a map with MODE=IN, only one data byte is reserved for each input field. This byte is set to X'00', and remains unchanged if the field is not selected. If the field is selected, the byte is set to X'FF'.

No other data is supplied, even if the field is a selection field and the ENTER key has been pressed.

If the data in a detectable field is required, all the following conditions must be fulfilled:

1. The field must begin with one of the following characters:

   ```
   ? > & blank
   ```

   and DET must be specified in the output map.
2. The ENTER key (or some other attention key) must be pressed after the field has been selected, although the ENTER key is not required for detectable fields beginning with & or a blank.
3. DET must not be specified for the field in the input map. DET must, however, be specified in the output map. For more information about BMS support of the light pen, see the *CICS Application Programming Guide*.

DRK specifies that the field is nonprint/nondisplay. DRK cannot be specified if DET is specified.

FSET specifies that the modified data tag (MDT) for this field must be set when the field is sent to a terminal.

Specification of FSET causes the 3270 to treat the field as though it has been modified. On a subsequent read from the terminal, this field is read, whether it has been modified. The MDT remains set until the field is rewritten without ATTRB=FSET, or until an output mapping request causes the MDT to be reset.

Either of two sets of defaults can apply when a field to be displayed on a 3270 is being defined but not all parameters are specified. If no ATTRB parameters are specified, ASKIP and NORM are assumed. If any parameter is specified, UNPROT and NORM are assumed for that field unless overridden by a specified parameter.

IC specifies that the cursor is to be placed in the first position of the field. The IC attribute for the last field for which it is specified in a map is the one that takes effect. If not specified for any fields in a map, the default location is zero. Specifying IC with ASKIP or PROT causes the cursor to be placed in an unkeyable field.

This option can be overridden by the CURSOR option of the SEND MAP command that causes the write operation.

NORM
specifies that the field intensity is to be normal.

NUM ensures that the data entry keyboard is set to numeric shift for this field unless the operator presses the alpha shift key, and prevents entry of nonnumeric data if the Keyboard Numeric Lock feature is installed.

PROT
specifies that data cannot be keyed into the field.

If data is to be copied from one device to another attached to the same 3270 control unit, the first position (address 0) in the buffer of the device to be copied from must not contain an attribute byte for a protected field. Therefore, when preparing maps for 3270s, ensure that the first map of any page does not contain a protected field starting at position 0.

UNPROT
specifies that data can be keyed into the field.

CASE
specifies that the field contains both uppercase and lowercase data that is to be converted to uppercase if the terminal definition specifies katakana support (KATAKANA(YES) option on RDO TYPETERM definition).

This must be specified if a field is known to contain lowercase Latin characters but can be displayed on a katakana display. It must not be specified if the field can contain valid katakana characters.

COLOR
indicates the individual color, or the default color for the mapset (where applicable).

The valid colors are blue, red, pink, green, turquoise, yellow, and neutral.

The COLOR operand is ignored unless the terminal supports color, as indicated by the RDO option COLOR.

GINIT
specifies constant or default data for an output field. GINIT is used to specify data in DBCS character strings, which must be enclosed by SO (shift out, X'0E') and SI (shift in, X'0F') characters. When GINIT is specified, the length must be even and is the number of bytes in the string (that is, not the number of DBCS characters). If a graphic data type (PS=X'F8') is used, and the language is stated as COBOL2 (Enterprise COBOL or VS COBOL II), a PIC G is generated. Only one of GINIT, INITIAL, or XINIT can be specified.

GRPNAME
is the name used to generate symbolic storage definitions and to combine specific fields under one group name. The same group name must be specified for each field that is to belong to the group. The length of the name is up to 30 characters, refer to your compiler manual to ensure that there are no other restrictions on the length.

The rules for defining group names are the same as for defining field names. See for details.

If this operand is specified, the OCCURS operand cannot be specified.

The fields in a group must follow on; there can be gaps between them, but not other fields from outside the group. A field name must be specified for every field that belongs to the group, and the POS operand must also be specified to ensure that the fields follow each other. All the DFHMDF macros defining the fields of a group must be placed together, and in the correct order (ascending numeric order of the POS value).

For example, the first 20 columns of the first six lines of a map can be defined as a group of six fields, as long as the remaining columns on the first five lines are not defined as fields.

The ATTRB operand specified on the first field of the group applies to all of the fields within the group.

HILIGHT
specifies the default highlighting attribute for all fields in all maps in a mapset.

OFF is the default and indicates that no highlighting is used.

BLINK
specifies that the field must flash.

REVERSE
specifies that the character or field is displayed in reverse video, for example, on a 3278, black characters on a green background.

UNDERLINE
specifies that a field is underlined.

The HILIGHT operand is ignored unless the terminal supports highlighting, as indicated by the RDO TYPETERM option HILIGHT(YES).

INITIAL (or XINIT)
specifies constant or default data for an output field. INITIAL is used to specify data in character form; XINIT is used to specify data in hexadecimal form.

For fields with the DET attribute, initial data that begins with one of the following characters must be supplied:

```
? > & blank
```

The number of characters that can be specified in the INITIAL operand is restricted to the continuation limitation of the assembler to be used or to the value specified in the LENGTH operand (whichever is the smaller).

Hexadecimal data is written as an even number of hexadecimal digits, for example, XINIT=C1C2. If the number of valid characters is smaller than the field length, the data is padded on the right with blanks. For example, if LENGTH=3, XINIT=C1C2 results in an initial field of 'AB'.

If hexadecimal data is specified that corresponds with line or format control characters, the results are unpredictable. The XINIT operand must therefore be used with care. Only one of GINIT, INITIAL, or XINIT can be specified.

JUSTIFY
specifies the field justifications for input operations. This operand is ignored for VTAM-supported 3600, 3650, and 3790 terminals, because input mapping is not available.

LEFT specifies that data in the input field is left-adjusted.

RIGHT
specifies that data in the input field is right-adjusted.

BLANK
specifies that blanks are to be inserted in any unfilled positions in an input field.

ZERO
specifies that zeros are to be inserted in any unfilled positions in an input field.

LEFT and RIGHT are mutually exclusive, as are BLANK and ZERO. If certain arguments are supplied but others are not, assumptions are made as follows:

Specified	Assumed
LEFT	BLANK
RIGHT	ZERO
BLANK	LEFT
ZERO	RIGHT

If JUSTIFY is omitted, but the NUM attribute is specified, RIGHT and ZERO are assumed. If JUSTIFY is omitted, but attributes other than NUM are specified, LEFT and BLANK are assumed.

Note: If a field is initialized by an output map or contains data from any other source, data that is typed as input overwrites only the equivalent length of the existing data; surplus existing data remains in the field and could cause unexpected interpretation of the new data.

LENGTH
specifies the length (1–256 bytes) of the field or group of fields. This length is the maximum length required for application program data to be entered into the field; it does not include the 1 byte attribute indicator appended to the field by CICS for use in subsequent processing. The length of each individual subfield within a group must not exceed 256 bytes.

In general LENGTH can be omitted if PICIN or PICOUT is specified, unless PICOUT defines a COBOL picture containing a currency symbol that replaces a currency sign of length greater than 1. LENGTH is required otherwise. You can specify a length of zero only if you omit the label (field name) from the DFHMDF macro. That is, the field is not part of the application data structure

and the application program cannot modify the attributes of the field. You can use a field with zero length to delimit an input field on a map.

The map dimensions specified in the SIZE operand of the DFHMDI macro defining a map can be smaller than the actual page size or screen size defined for the terminal.

If the LENGTH specification in a DFHMDF macro causes the map-defined boundary on the same line to be exceeded, the field on the output screen is continued by wrapping.

OCCURS
specifies that the indicated number of entries for the field are to be generated in a map, and that the map definition is to be generated in such a way that the fields are addressable as entries in a matrix or an array. This permits several data fields to be addressed by the same name (subscripted) without generating a unique name for each field.

OCCURS and GRPNAME are mutually exclusive; that is, OCCURS cannot be used when fields have been defined under a group name. If this operand is omitted, a value of OCCURS=1 is assumed.

OUTLINE
allows lines to be included above, below, to the left, or to the right of a field. You can use these lines in any combination to construct boxes around fields or groups of fields.

PICIN (COBOL and PL/I only)
specifies a picture to be applied to an input field in an IN or INOUT map; this picture serves as an editing specification that is passed to the application program, thus permitting the user to use the editing capabilities of COBOL or PL/I. BMS checks that the specified characters are valid picture specifications for the language of the map.

However, the validity of the input data is not checked by BMS or the high-level language when the map is used, so any wanted checking must be performed by the application program. The length of the data associated with "value" must be the same as that specified in the LENGTH operand if LENGTH is specified. If both PICIN and PICOUT are used, an error message is produced if their calculated lengths do not agree; the shorter of the two lengths is used. If PICIN or PICOUT is not coded for the field definition, a character definition of the field is automatically generated regardless of other operands that are coded, such as ATTRB=NUM.

As an example, assume that the following map definition is created for reference by a COBOL application program:

```
MAPX  DFHMSD  TYPE=DSECT,
              LANG=COBOL,
              MODE=INOUT
MAP   DFHMDI  LINE=1,COLUMN=1,
              SIZE=(1,80)
F1    DFHMDF  POS=0,LENGTH=30
F2    DFHMDF  POS=40,LENGTH=10,
              PICOUT='$$$,$$0.00'
F3    DFHMDF  POS=60,LENGTH=6,
              PICIN='9999V99',
              PICOUT='ZZ9.99'
      DFHMSD  TYPE=FINAL
```

This generates the following DSECT:

```
01  MAPI.
    02  F1L     PIC S9(4) COMP.
    02  F1A     PIC X.
    02  FILLER REDEFINES F1A.
     03  F1F    PIC X.
    02  F1I     PIC X(30).
    02  FILLER  PIC X.
    02  F2L     PIC S9(4) COMP.
    02  F2A     PIC X.
    02  FILLER REDEFINES F2A.
     03  F2F    PIC X.
    02  F2I     PIC X(10).
    02  FILLER  PIC X.
    02  F3L     PIC S9(4) COMP.
    02  F3A     PIC X.
    02  FILLER REDEFINES F3A.
     03  F3F    PIC X.
    02  F3I     PIC 9999V99.
    02  FILLER  PIC X.
01  MAPO REDEFINES MAPI.
    02  FILLER  PIC X(3).
    02  F1O     PIC X(30).
    02  FILLER  PIC X.
    02  FILLER  PIC X(3).
    02  F2O     PIC $$$,$$0.00.
    02  FILLER  PIC X.
    02  FILLER  PIC X(3).
    02  F3O     PIC ZZ9.99.
    02  FILLER  PIC X.
```

Valid picture values for COBOL input maps are:

```
A P S V X 9 / and (
```

Valid picture values for PL/I input maps are:

```
A B E F G H I K M P R S T V
X Y and Z

1 2 3 6 7 8 9 / + - , . *
$ and (
```

Refer to the appropriate language reference manual for the correct syntax of the PICTURE attribute.

Note: PL/I supports multiple currency signs and multi-character currency signs in PICTURE specifications.

The default currency picture symbol is the dollar sign ($), which represents the national currency symbol; for example the dollar ($), the pound (£), or the yen (¥).

The default currency picture symbol can be replaced by a currency string enclosed by less than (<) and greater than (>) symbols. For example:

```
DECLARE
    USPRICE PICTURE '$99.99',
    UKPRICE PICTURE '<£>99.99',
    EUPRICE PICTURE '<EUR>99.99';
```

PICOUT (COBOL and PL/I only)
is like PICIN, except that a picture to be applied to an output field in the OUT or INOUT map is generated.

Valid picture values for COBOL output maps are:

```
A B E P S V X Z 0 9 , . + - $
CR DB / and (
```

Valid picture values for PL/I output maps are:

```
A B E F G H I K M P R S T V
X Y and Z

1 2 3 6 7 8 9 / + - , . * $
CR DB and (
```

Refer to the appropriate language reference manual for the correct syntax of the PICTURE attribute.

Note: PL/I supports multiple currency signs and multi-character currency signs in PICTURE specifications.

The default currency picture symbol is the dollar sign ($), which represents the national currency symbol; for example the dollar ($), the pound (£), or the yen (¥).

The default currency picture symbol can be replaced by a currency string enclosed by less than (<) and greater than (>) symbols. For example:

```
DECLARE
    USPRICE PICTURE '$99.99',
    UKPRICE PICTURE '<£>99.99',
    EUPRICE PICTURE '<EUR>99.99';
```

Note: COBOL supports multiple currency signs and multi-character currency signs in PICTURE specifications.

The default currency picture symbol is the dollar sign ($), which represents the national currency symbol; for example the dollar ($), the pound (£), or the yen (¥).

The default currency picture symbol can be replaced by a different currency picture symbol that is defined in the SPECIAL NAMES clause. The currency sign represented by the picture symbol is defined in the same clause. For example:

```
SPECIAL NAMES.
CURRENCY SIGN IS '$' WITH PICTURE SYMBOL '$'.
CURRENCY SIGN IS '£' WITH PICTURE SYMBOL '£'.
CURRENCY SIGN IS 'EUR' WITH PICTURE SYMBOL '#'.

WORKING STORAGE SECTION.
01 USPRICE PIC $99.99.
01 UKPRICE PIC £99.99.
01 ECPRICE PIC #99.99.
```

LENGTH must be specified when PICOUT specifies a COBOL picture containing a currency symbol that replaces a currency sign of length greater than 1.

POS

specifies the location of a field. This operand specifies the individually addressable character location in a map at which the attribute byte that precedes the field is positioned.

number

specifies the displacement (relative to zero) from the beginning of the map being defined.

(line,column)

specify lines and columns (relative to one) within the map being defined.

The location of data on the output medium is also dependent on DFHMDI operands.

The first position of a field is reserved for an attribute byte. When supplying data for input mapping from non-3270 devices, the input data must allow space for this attribute byte. Input data must not start in column 1 but can start in column 2.

The POS operand always contains the location of the first position in a field, which is normally the attribute byte when communicating with the 3270. For the second and subsequent fields of a group, the POS operand points to an assumed attribute-byte position, ahead of the start of the data, even though no actual attribute byte is necessary. If the fields follow on immediately from one another, the POS operand points to the last character position in the previous field in the group.

When a position number is specified that represents the last character position in the 3270, 2 special rules apply:

- ATTRIB=IC must not be coded. The cursor can be set to location zero by using the CURSOR option of a SEND MAP, SEND CONTROL, or SEND TEXT command.
- If the field is to be used in an output mapping operation with MAP=DATAONLY on the SEND MAP command, an attribute byte for that field must be supplied in the symbolic map data structure by the application program.

PS

specifies that programmed symbols are to be used. This overrides any PS operand set by the DFHMDI macro or the DFHMSD macro.

BASE is the default and specifies that the base symbol set is to be used.

psid specifies a single EBCDIC character, or a hexadecimal code of the form X'nn', that identifies the set of programmed symbols to be used.

The PS operand is ignored unless the terminal supports programmed symbols, as indicated by PROGSYMBOLS(YES) on the RDO TYPETERM definition.

SOSI

indicates that the field can contain a mixture of EBCDIC and DBCS data. The DBCS subfields within an EBCDIC field are delimited by SO (shift out) and SI (shift in) characters. SO and SI both occupy a single screen position (normally displayed as a blank). They can be included in any non-DBCS field on output, if they are correctly paired. The terminal user can transmit them inbound if they are already present in the field, but can add them to an EBCDIC field only if the field has the SOSI attribute.

TRANSP

determines whether the background of an alphanumeric field is transparent or opaque, that is, whether an underlying (graphic) presentation space is visible between the characters.

VALIDN

specifies that:

- validation is to be used on an 8775 terminal
- this field can be processed by the BMS global user exits

This overrides any VALIDN operand on the DFHMDI macro or the DFHMSD macro.

MUSTFILL
specifies that the field must be filled completely with data. An attempt to move the cursor from the field before it has been filled, or to transmit data from an incomplete field, raises the INHIBIT INPUT condition

MUSTENTER
specifies that data must be entered into the field, though need not fill it. An attempt to move the cursor from an empty field raises the INHIBIT INPUT condition

TRIGGER
specifies that this field is a trigger field. Trigger fields are discussed in Trigger field support.

USEREXIT
specifies that this field is to be processed by the BMS global user exits, XBMIN and XBMOUT, if this field is received or transmitted in a 3270 datastream when the respective exit is enabled.

The MUSTFILL, MUSTENTER, and TRIGGER specifications are valid only for terminals that support the field validation extended attribute, otherwise they are ignored. The USEREXIT specification applies to all 3270 devices.

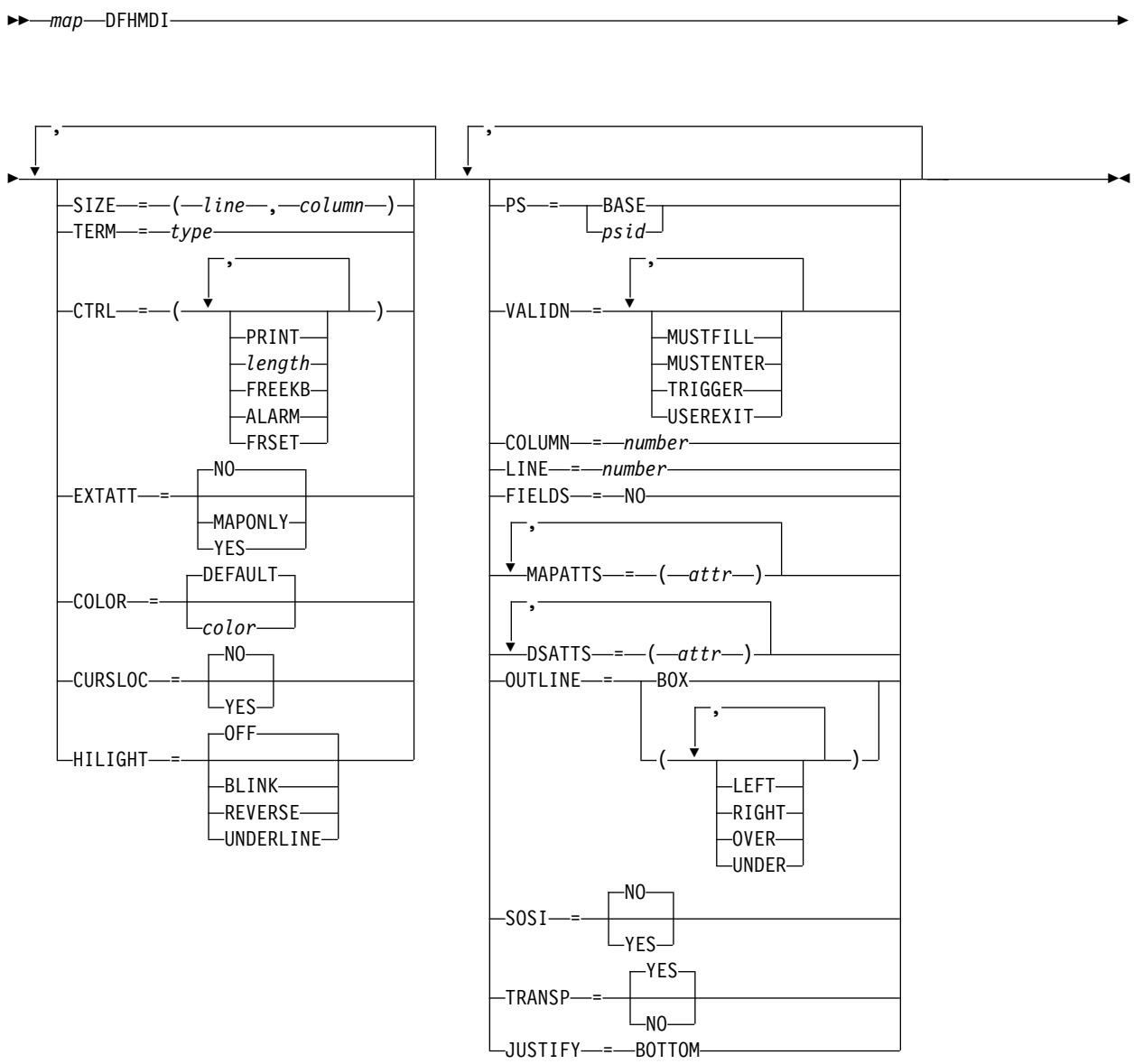
Note: The USEREXIT specification is unconnected with the field validation extended attribute as defined in the 3270 datastream architecture.

XINIT
see INITIAL, earlier in the list. Only one of GINIT, INITIAL, or XINIT can be specified.

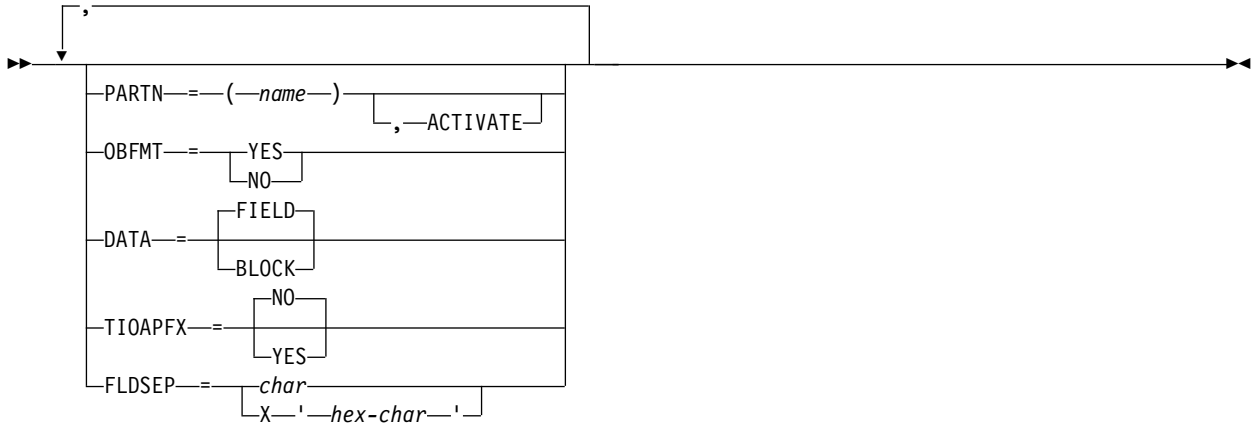
DFHMDI

The DFHMDI macro defines a map within the mapset defined by a previous DFHMSD macro.

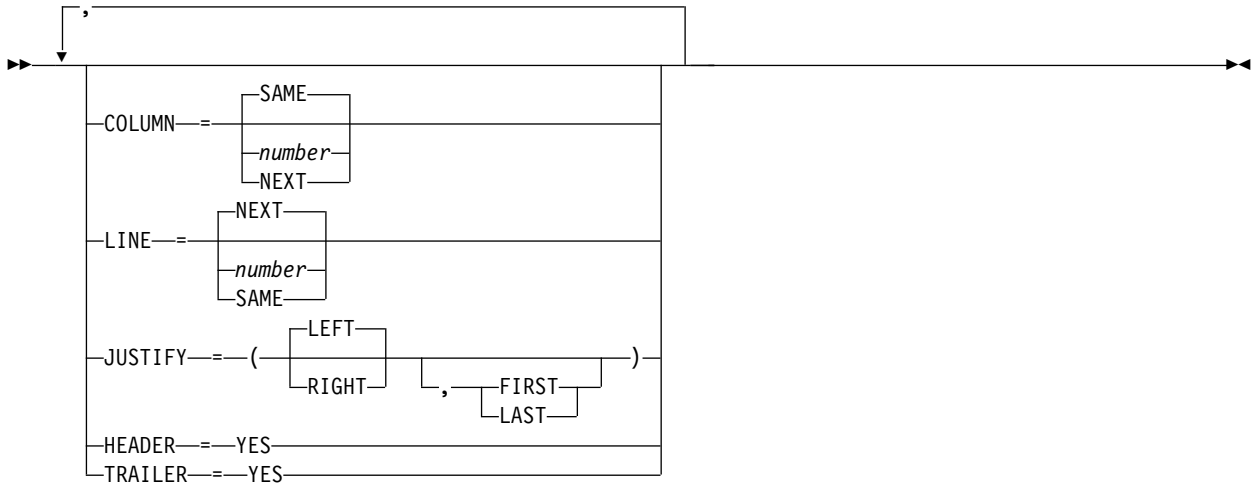
DFHMDI Minimum BMS



DFHMDI Standard BMS



DFHMDI Full BMS



A map contains zero or more fields.

"map" is the name (1–7 characters) of the map.

Operands

COLOR
indicates the individual color, or the default color for the mapset (where applicable). This is overridden by the COLOR operand of the DFHMDF macro.

The valid colors are blue, red, pink, green, turquoise, yellow, and neutral.

The COLOR operand is ignored unless the terminal supports color, as indicated by the RDO TYPETERM definition option COLOR(YES).

COLUMN

specifies the column in a line at which the map is to be placed, that is, it establishes the left or right map margin. The JUSTIFY operand of the DFHMDI macro controls whether map and page margin selection and column counting are to be from the left or right side of the page. The columns between the specified map margin and the page margin are not available for subsequent use on the page for any lines included in the map.

NUMBER

is the column from the left or right page margin where the left or right map margin is to be established.

NEXT

indicates that the left or right map margin is to be placed in the next available column from the left or right on the current line.

SAME

indicates that the left or right map margin is to be established in the same column as the last nonheader or nontrailer map used that specified COLUMN=number and the same JUSTIFY operands as this macro.

For input operations, the map is positioned at the extreme left or right side, depending on whether JUSTIFY=LEFT or JUSTIFY=RIGHT has been specified.

CTRL

defines characteristics of IBM 3270 terminals. Use of any of the control options in the SEND MAP command overrides all control options in the DFHMDI macro, which in turn overrides all control options in the DFHMSD macro.

If CTRL is used with cumulative BMS paging (that is, the ACCUM option is used on the BMS SEND MAP commands), it must be specified on the last (or only) map of a page, unless it is overridden by the ALARM, FREEKB, and so on, options on the SEND MAP or accumulated SEND CONTROL command.

PRINT

must be specified if the printer is to be started; if omitted, the data is sent to the printer buffer but is not printed. This operand is ignored if the mapset is used with 3270 displays without the Printer Adapter feature.

LENGTH

indicates the line length on the printer; length can be specified as L40, L64, L80, or HONEOM. L40, L64, and L80 force a new line after 40, 64, or 80 characters. HONEOM causes the default printer line length to be used. If this option is omitted, BMS sets the line length from the TCT page size.

FREEKB

causes the keyboard to be unlocked after the map is written. If FREEKB is not specified, the keyboard remains locked; data entry from the keyboard is inhibited until this status is changed.

ALARM

activates the 3270 audible alarm. For non-3270 VTAM terminals it sets the alarm flag in the FMH. (This feature is not supported by interactive and batch logical units.)

FRSET

specifies that the modified data tags (MDTs) of all fields currently in

the 3270 buffer are to be reset to an unmodified condition (that is, field reset) before map data is written to the buffer. This allows the DFHMDF macro with the ATTRB operand to control the final status of any fields written or rewritten in response to a BMS command.

Note: CTRL cannot be specified in the DFHMDI and DFHMSD macros in the same mapset.

CURSLOC
indicates that for all RECEIVE MAP operations using this map on 3270 terminals, BMS sets a flag in the application data structure element for the field where the cursor is located.

The flag might be tested by DFHBMCUR (see copybook DFHBMSCA in Chapter 9, "BMS-related constants," on page 849).

To test the flag (COBOL example):

```
(DFHBMSCA)
...
02  DFHBMEOF   PIC X VALUE X'80'.
02  DFHBMCUR   PIC X VALUE X'02'.
02  DFHBMEC    PIC X VALUE X'82'.
02  DFHBMFLG   PIC X.
  88  DFHERASE       VALUES ARE X'80', X'82'.
  88  DFHCURSR       VALUES ARE X'02', X'82'.
MOVE FLD1F TO DFHBMFLG.
IF DFHERASE THEN ...
            ELSE ...
IF DFHCURSR THEN ...
            ELSE ...
```

Note:

1. If CURSLOC=YES is specified for the MAP definitions, and there is no data for any field of the application data structure, but the cursor lies within a field known to the application data structure, BMS sets the cursor flag for the appropriate field, but the data for all fields in the application data structure is null, and the MAPFAIL condition does not occur. The unmapped data stream is not available to the application program unless it is a RECEIVE DATA FROM request.
2. A valid CURSLOC definition in DFHMDI overrides the definition in DFHMSD.

DATA
specifies the format of the data.

FIELD
specifies that the data is passed as contiguous fields, each field having the format:

LL	A	data field

"LL" is two bytes specifying the length of the data as input from the terminal (ignored in output processing). "A" is a byte into which the programmer can place an attribute to override that specified in the map used to process this data (see copybook DFHBMSCA in Chapter 9, "BMS-related constants," on page 849).

BLOCK
specifies that the data is passed as a continuous stream in the following format:

A	data field	space

This stream is processed as line segments of the length specified in the map used to process the data. The data is in the form in which it appears at the terminal; that is, it contains data fields and interspersed blanks corresponding to any spaces that are to appear between the fields on output. You cannot use DSATTS=YES if you specify DATA=BLOCK.

Block data is further discussed in Block data .

DSATTS
specifies the attribute types to be included in the symbolic description map. These types can be one or more of the following: COLOR, HILIGHT, OUTLINE, PS, SOSI, TRANSP, and VALIDN. Any type included in DSATTS should also be included in MAPATTS.

EXTATT
this operand is supported for compatibility with previous releases. Each of the extended attributes can be defined individually. For new maps, the operands DSATTS and MAPATTS must be used instead.

NO is equivalent to not specifying the DSATTS operand or the MAPATTS operand.

YES is equivalent to:

```
MAPATTS=(COLOR,HILIGHT,PS,VALIDN)
DSATTS=(COLOR,HILIGHT,PS,VALIDN)
```

MAPONLY
is equivalent to:

```
MAPATTS=(COLOR,HILIGHT,PS,VALIDN)
```

FIELDS
specifies whether the map contains fields. If you specify FIELDS=NO, you create a null map that defines a “hole” in BMS's view of the screen. BMS cannot change the contents of such a hole after it has created it by sending a null map.

FLDSEP
specifies the field separator sequence (1–4 characters) for input from non-3270 devices. Input from non-3270 devices can be entered as a single string of data with the field separator sequence delimiting fields. The data between the field separators is moved to the input fields in the map in order.

HEADER
allows the map to be used during page building without terminating the OVERFLOW condition. This operand can be specified for more than one map in a map set.

HILIGHT
specifies the default highlighting attribute for all fields in all maps in a mapset. This is overridden by the HILIGHT operand of the DFHMDF.

OFF is the default and indicates that no highlighting is used.

BLINK
specifies that the field must flash.

REVERSE
specifies that the character or field is displayed in reverse video, for example, on a 3278, black characters on a green background.

UNDERLINE
specifies that a field is underlined.

The HILIGHT operand is ignored unless the terminal supports highlighting, as indicated by HILIGHT(YES) on the RDO TYPETERM definition,

JUSTIFY
specifies the position of the map on the page.

LEFT specifies that the map is to be positioned starting at the specified column from the left margin on the specified line.

RIGHT
specifies that the map is to be positioned starting at the specified column from the right margin on the specified line.

FIRST
specifies that the map is to be positioned as the first map on a new page. Any partially formatted page from preceding BMS commands is considered to be complete. This operand can be specified for only one map per page.

LAST indicates that the map is to be positioned at the foot of the current page. This operand can be specified for multiple maps to be placed on one page. However, maps other than the first map for which it is specified must be able to be positioned horizontally without requiring that more lines be used.

BOTTOM
for a SEND MAP ACCUM command has the same effect as LAST. For a SEND MAP command (without ACCUM) and a RECEIVE MAP command, JUSTIFY=BOTTOM positions the map at the foot of the screen if the number of lines in the map is specified in the SIZE operand. No account is taken of trailer maps in the mapset. JUSTIFY=BOTTOM is equivalent to specifying

```
LINE=(screendepth-mapdepth+1)
```

on the map definition, but it allows the same map to be used for different screen sizes. JUSTIFY=BOTTOM is ignored if the number of lines is not also specified. If JUSTIFY=BOTTOM and LINE are both specified, the value specified in LINE is ignored.

LEFT and RIGHT are mutually exclusive, as are FIRST and LAST. If FIRST or LAST are not specified, the data is mapped at the next available position as determined by other parameters of the map definition and the current mapping operation. FIRST or LAST is ignored unless ACCUM is specified on SEND MAP commands; otherwise only one map is placed on each page.

Note: If a field is initialized by an output map or contains data from any other source, data that is keyed as input overwrites only the equivalent length of the existing data; surplus existing data remains in the field and could cause unexpected interpretation of the new data.

LINE
specifies the starting line on a page in which data for a map is to be formatted.

NUMBER
is a value in the range 1–240, specifying a starting line number. A request to map, on a line and column, data that has been formatted in

response to a preceding BMS command, causes the current page to be treated as though complete. The new data is formatted at the requested line and column on a new page.

NEXT
specifies that formatting of data is to begin on the next available empty line. If LINE=NEXT is specified in the DFHMDI macro, it is ignored for input operations and LINE=1 is assumed.

SAME
specifies that formatting of data is to begin on the same line as that used for a preceding BMS command. If COLUMN=NEXT is specified, it is ignored for input operations and COLUMN=1 is assumed. If the data does not fit on the same line, it is placed on the next available line that is empty.

MAPATTS
specifies the attribute types to be included in the physical map. These types can be one or more of the following: COLOR, HILIGHT, OUTLINE, PS, SOSI, TRANSP, and VALIDN. This list must include all the attribute types to be specified for individual fields in the map (DFHMDF macro).

Where possible these values are deduced from operands already specified in the DFHMDI and DFHMSD macros. For example, if COLOR=BLUE has been specified, MAPATTS=COLOR is assumed.

OBFMT
specifies whether outboard formatting is to be used. This operand is available only for 3650 logical units, or for an 8100 series processor running DPS Release 2 and defined to CICS as an LUTYPE2 logical unit. For more information, see Outboard formatting.

The OBFMT operand overrides the OBFMT operand on the DFHMSD macro.

YES specifies that this map definition can be used in outboard formatting.

NO specifies that this map definition cannot be used in outboard formatting.

OUTLINE
allows lines to be included above, below, to the left, or to the right of a field. You can use these lines in any combination to construct boxes around fields or groups of fields.

PARTN
specifies the default partition to be associated with maps in this mapset. If the ACTIVATE option is specified, the specified partition is also activated when maps in this mapset are output to a terminal that supports partitions.

This option overrides the PARTN option of the DFHMSD macro and is overridden by any OUTPARTN or ACTPARTN option on the SEND MAP command, or the INPARTN option on a RECEIVE MAP command.

The PARTN option is ignored if the target terminal does not support partitions, or if there is no partition set associated with the transaction.

PS
specifies that programmed symbols are to be used. This overrides the PS operand of the DFHMSD macro and is overridden by the PS operand of the DFHMDF macro.

BASE specifies that the base symbol set is to be used.

psid specifies a single EBCDIC character, or a hexadecimal code of the form X'nn', that identifies the set of programmed symbols to be used.

The PS operand is ignored unless the terminal supports programmed symbols, as indicated by the PROGSYMBOLS(YES) on the RDO TYPETERM definition.

SIZE
specifies the size of a map.

line is a value in the range 1–240, specifying the depth of a map as a number of lines.

column
is a value in the range 1–240, specifying the width of a map as a number of columns.

This operand is required in the following cases:

- An associated DFHMDF macro with the POS operand is used.
- The map is to be referred to in a SEND MAP command with the ACCUM option.
- The map is to be used when referring to input data from other than a 3270 terminal in a RECEIVE MAP command.
- The map is to be used to send or receive data by way of the CICS 3270 Web Bridge.

SOSI
indicates that the field can contain a mixture of EBCDIC and DBCS data. The DBCS subfields within an EBCDIC field are delimited by SO (shift out) and SI (shift in) characters. SO and SI both occupy a single screen position (normally displayed as a blank). They can be included in any non-DBCS field on output, if they are correctly paired. The terminal user can transmit them inbound if they are already present in the field, but can add them to an EBCDIC field only if the field has the SOSI attribute.

TERM
kept for compatibility with previous releases.

TIOAPFX
specifies whether BMS should include a filler in the symbolic description maps to allow for the unused TIOA prefix. This operand overrides the TIOAPFX operand specified or defaulted for the DFHMSD macro. If it is not specified, the value specified or defaulted on the DFHMSD macro is used.

YES specifies that the filler should be included in the symbolic description maps. Always use TIOAPFX=YES for command-level application programs.

NO specifies that the filler is not to be included.

TRAILER
allows the map to be used during page building without terminating the OVERFLOW condition. This operand can be specified for more than one map in a mapset. If a trailer map is used other than in the overflow environment, the space normally reserved for overflow trailer maps is not reserved while mapping the trailer map.

TRANSP
determines whether the background of an alphanumeric field is transparent or opaque, that is, whether an underlying (graphic) presentation space is visible between the characters.

VALIDN

specifies that:

- validation is to be used on an 8775 terminal
- this field can be processed by the BMS global user exits

This is overridden by the VALIDN operand of the DFHMDF macro, and overrides the VALIDN operand of the DFHMSD macro.'

MUSTFILL

specifies that the field must be filled completely with data. An attempt to move the cursor from the field before it has been filled, or to transmit data from an incomplete field, raises the INHIBIT INPUT condition.

MUSTENTER

specifies that data must be entered into the field, though need not fill it. An attempt to move the cursor from an empty field raises the INHIBIT INPUT condition.

TRIGGER

specifies that this field is a trigger field. Trigger fields are discussed in Trigger field support.

USEREXIT

specifies that this field is to be processed by the BMS global user exits, XBMIN and XBMOUT, if this field is received or transmitted in a 3270 datastream when the respective exit is enabled.

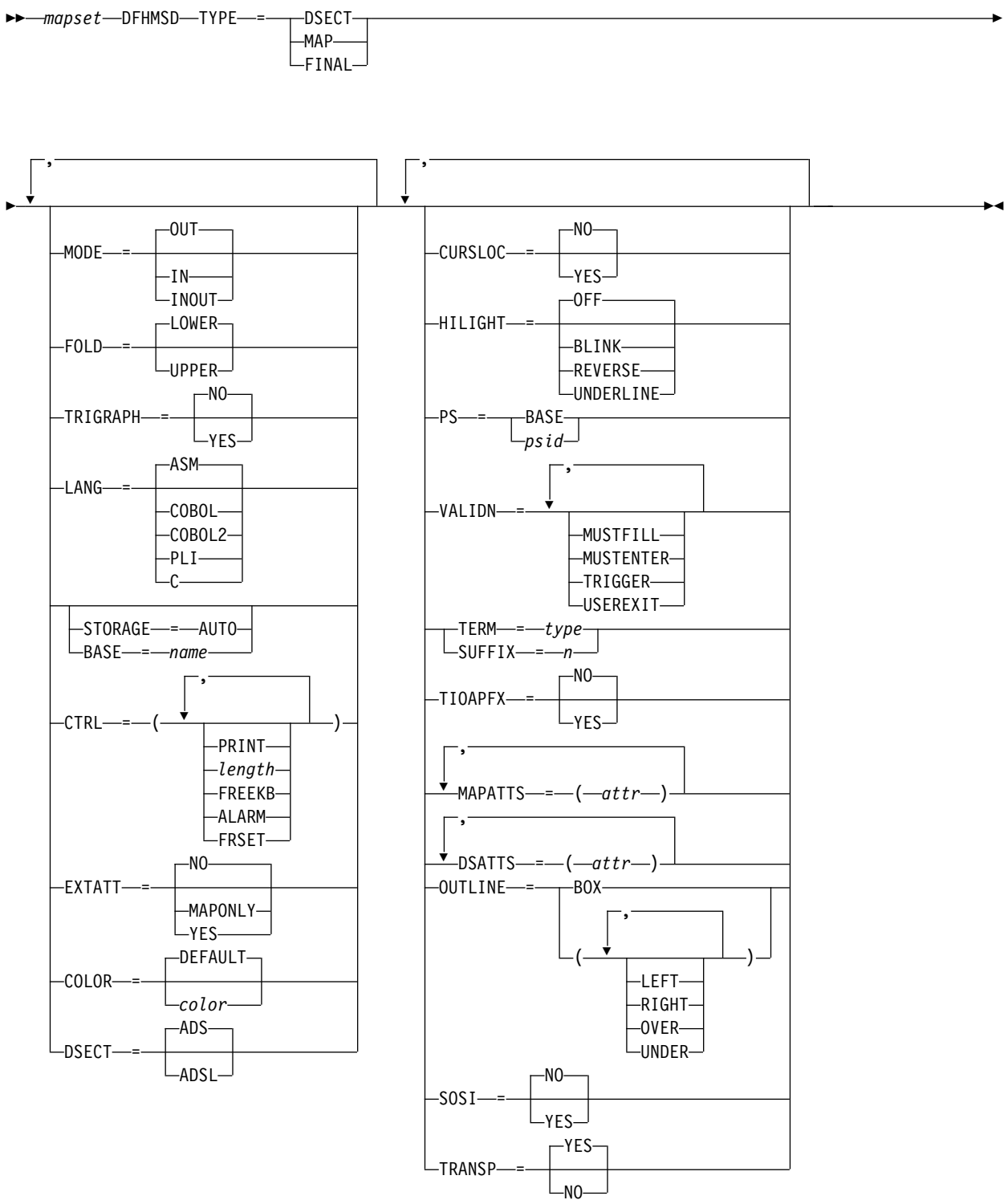
The MUSTFILL, MUSTENTER and TRIGGER specifications are valid only for terminals that support the field validation extended attribute, otherwise they are ignored. The USEREXIT specification applies to all 3270 devices.

Note: The USEREXIT specification is totally unconnected with the field validation extended attribute as defined in the 3270 datastream architecture.

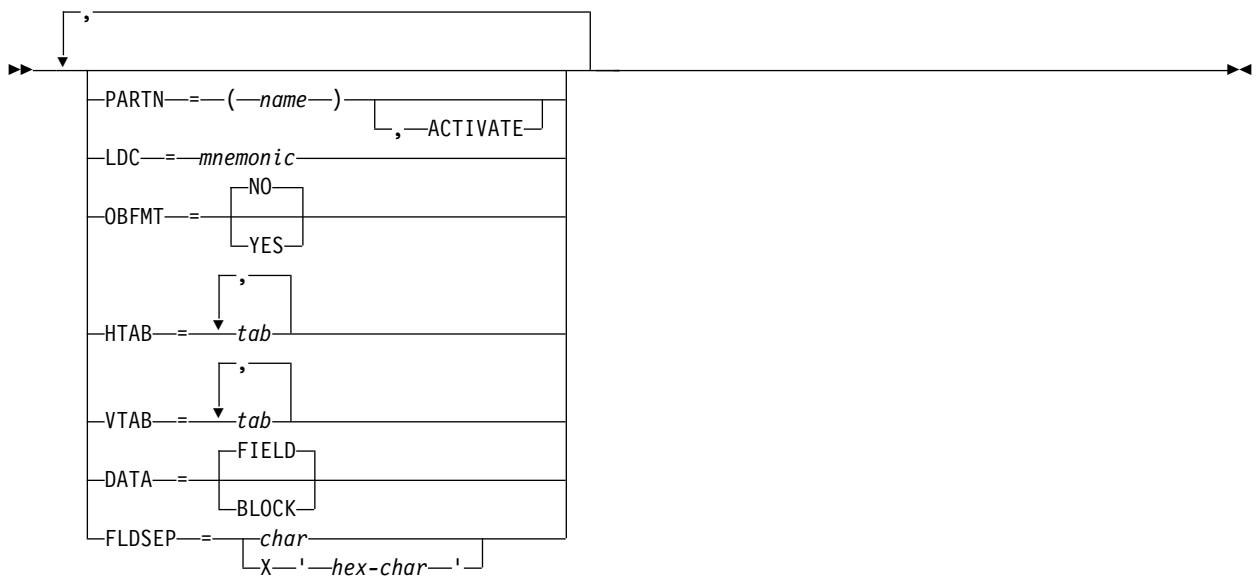
DFHMSD

The DFHMSD macro defines a mapset.

DFHMSD Minimum BMS



DFHMSD Standard BMS



A DFHMSD macro defines a mapset. It begins with the following code:

```
DFHMSD TYPE=MAP    (or TYPE=DSECT)
```

It ends with the following code:

```
DFHMSD TYPE=FINAL
```

mapset is the name of the mapset. Normally, the name is up to 7 characters in length. However, if the mapset is used to generate HTML templates, and contains more than 36 maps, the name must not exceed 6 characters in length.

A DFHMSD macro contains one or more map definition macros, each of which contains one or more field definition macros.

Ensure that the names of maps, and names of fields in a mapset (or in multiple mapsets that are copied into one application program) are unique. However, a map can have the same name as a mapset.

Before CICS can load a physical map, you must define a physical map using an RDO **DEFINE MAPSET** command.

You assemble a BMS mapset definition to generate either a symbolic description map or a physical map. For information about assembling and cataloging the maps, see .

Operands

BASE

Specifies that the same storage base is used for the symbolic description maps from more than one mapset. The same name is specified for each mapset that is to share the same storage base. Because all mapsets with the same base describe the same storage, data related to a previously used mapset might be overwritten when a new mapset is used. Different maps in the same mapset also overlay one another.

This operand is not valid for assembler language programs, and cannot be used when STORAGE=AUTO has been specified.

COLOR

Indicates the individual color, or the default color for the mapset (where applicable). This is overridden by the COLOR operand of the DFHMDI macro, which is in turn overridden by the COLOR operand of the DFHMDF macro.

The valid colors are blue, red, pink, green, turquoise, yellow, and neutral.

The COLOR operand is ignored unless the terminal supports color, as indicated by the RDO TYPETERM definition COLOR(YES) option.

CTRL

Defines characteristics of IBM 3270 terminals. Use of **any** of the control options in the SEND MAP command overrides all control options in the DFHMDI macro, which in turn overrides all control options in the DFHMSD macro.

If CTRL is used with cumulative BMS paging (that is, the ACCUM option is used on the BMS SEND MAP commands), it must be specified on the last (or only) map of a page, unless it is overridden by the ALARM, FREEKB, and so on, options on the SEND MAP or accumulated SEND CONTROL command.

PRINT

Must be specified if the printer is to be started; if omitted, the data is sent to the printer buffer but is not printed. This operand is ignored if the mapset is used with 3270 displays without the Printer Adapter feature.

LENGTH

Indicates the line length on the printer; length can be specified as L40, L64, L80, or HONEOM. L40, L64, and L80 force a new line after 40, 64, or 80 characters, respectively. HONEOM causes the default printer line length to be used. If this option is omitted, BMS sets the line length from the TCT page size.

FREEKB

Causes the keyboard to be unlocked after the map is written. If FREEKB is not specified, the keyboard remains locked; data entry from the keyboard is inhibited until this status is changed.

ALARM

Activates the 3270 audible alarm. For non-3270 VTAM terminals, it sets the alarm flag in the FMH. (This feature is not supported by interactive and batch logical units.)

FRSET

Specifies that the modified data tags (MDTs) of all fields currently in the 3270 buffer are to be reset to a not-modified condition (that is, field reset) before map data is written to the buffer. This allows the DFHMDF macro with the ATTRB operand to control the final status of any fields written or rewritten in response to a BMS command.

CURSLOC

Indicates that for all RECEIVE MAP operations using this map on 3270 terminals, BMS sets a flag in the application data structure element for the field where the cursor is located.

The flag can be tested by DFHBMCUR (see copybook DFHBMSCA in Chapter 9, "BMS-related constants," on page 849).

To test the flag (COBOL example):

```
(DFHBMSCA)
...
02  DFHBMEOF   PIC X VALUE X'80'.
02  DFHBMCUR   PIC X VALUE X'02'.
02  DFHBMEC    PIC X VALUE X'82'.
02  DFHBMFLG   PIC X.
  88  DFHERASE       VALUES ARE X'80', X'82'.
  88  DFHCURSR       VALUES ARE X'02', X'82'.
MOVE FLD1F TO DFHBMFLG.
IF DFHERASE THEN ...
            ELSE ...
IF DFHCURSR THEN ...
            ELSE ...
```

Note:

1. If CURSLOC=YES is specified for the MAP definitions, and there is no data for any field of the application data structure, but the cursor lies in a field known to the application data structure, BMS sets the cursor flag for the appropriate field, but the data for all fields in the application data structure is null, and the MAPFAIL condition does not occur. The unmapped data stream is not available to the application program unless it is a RECEIVE DATA FROM request.
2. A valid CURSLOC definition in DFHMDI overrides the definition in DFHMSD.

DATA

Specifies the format of the data.

FIELD

Specifies that the data is passed as contiguous fields, each field having the format:

LL	A	data field

LL is two bytes that specify the length of the data as input from the terminal (these two bytes are ignored in output processing). A is a byte into which the programmer can place an attribute to override that specified in the map used to process this data (see copybook DFHBMSCA in Chapter 9, "BMS-related constants," on page 849).

BLOCK

Specifies that the data is passed as a continuous stream in the following format:

A	data field	space

This stream is processed as line segments of the length specified in the map used to process the data. The data is in the form that it appears on the terminal; that is, it contains data fields and interspersed blanks

corresponding to any spaces that are to appear between the fields on output. You cannot use DSATTS=YES, if you specify DATA=BLOCK.

Block data is further discussed in .

DSATTS
Specifies the attribute types to be included in the symbolic description map. These types can be one or more of the following: COLOR, HILIGHT, OUTLINE, PS, SOSI, TRANSP, and VALIDN. Any type included in DSATTS should also be included in MAPATTS.

DSECT
Specifies whether the copybook generated supports the normal or long form of the application data structure (ADS).

ADS Requests mapping of the normal form of the ADS (default).

ADSL
Requests mapping of the long form of the ADS, where all fields are aligned on four-byte boundaries. This form is required by the 3270 Bridge interface to IBM MQ.

This option requires LANG=C.

EXTATT
This operand is supported for compatibility with previous releases. Each extended attribute can be defined individually. For new maps, use the operands DSATTS and MAPATTS instead.

NO Is equivalent to specifying neither the DSATTS operand nor the MAPATTS operand.

YES Is equivalent to specifying the following operands:

```
MAPATTS=(COLOR,HILIGHT,PS,VALIDN)
DSATTS=(COLOR,HILIGHT,PS,VALIDN)
```

MAPONLY
Is equivalent to specifying the following operand:

```
MAPATTS=(COLOR,HILIGHT,PS,VALIDN)
```

FLDSEP
Specifies the field separator sequence (1–4 characters) for input from non-3270 devices. Input from non-3270 devices can be entered as a single string of data with the field separator sequence delimiting fields. The data between the field separators is moved to the input fields in the map in order.

FOLD
Specifies whether to generate lowercase or uppercase characters in C language programs.

FOLD is only available for programs written in C.

HILIGHT
Specifies the default highlighting attribute for all fields in all maps in a mapset. This is overridden by the HILIGHT operand of the DFHMDI, which is in turn overridden by the HILIGHT operand of the DFHMDF.

OFF Indicates that no highlighting is used (the default).

BLINK
Specifies that the field must flash.

REVERSE
Specifies that the character or field is displayed in reverse video, for example, on a 3278, black characters on a green background.

UNDERLINE
Specifies that a field is underlined.

The HILIGHT operand is ignored unless the terminal supports highlighting, as indicated by HILIGHT(YES) on the RDO TYPETERM definition.

HTAB
Specifies one or more tab positions for use with interactive and batch logical units and SCS printers with horizontal forms control.

LANG
Specifies the source language of the application programs into which the symbolic description maps in the mapset are copied. COBOL is OS/VS COBOL, which cannot run under this CICS version, and COBOL2 is either Enterprise COBOL or VS COBOL II. This option need only be coded for DFHMSD TYPE=DSECT. If a mapset is to be used by more than one program, and the programs are not all written in the same source language, a separate version of the mapset must be defined for each programming language.

LDC
Specifies the code to be used by CICS to determine the logical device mnemonic to be used for a BMS output operation. If no LDC operand has been specified on any previous BMS output in the logical message, this LDC will be transmitted in the function management header to the logical unit. This operand is used only for VTAM-supported 3600 terminals, and batch logical units.

MAPATTS
Specifies the attribute types to be included in the physical map. These types can be one or more of the following: COLOR, HILIGHT, OUTLINE, PS, SOSI, TRANSP, and VALIDN. This list must include all the attribute types to be specified for individual fields in the map (DFHMDF macro).

Where possible these values are deduced from operands already specified in the DFHMDI and DFHMSD macros. For example, if COLOR=BLUE has been specified, MAPATTS=COLOR is assumed.

MODE
Specifies whether the mapset is to be used for input, output, or both.

OBFMT
Specifies whether outboard formatting is to be used. This operand is available only for 3650 logical units, or for an 8100 series processor running DPS Release 2 and defined to CICS as an LUTYPE2 logical unit. For more information, see .

The OBFMT operand on DFHMSD is overridden by the OBFMT operand on DFHMDI.

YES Specifies that all maps in this mapset can be used in outboard formatting, except those for which OBFMT=NO is specified in the DFHMDI macro.

NO Specifies that no maps in this mapset can be used in outboard formatting, except those for which OBFMT=YES is specified in DFHMDI.

OUTLINE
Allows lines to be included above, below, to the left, or to the right of a field. You can use these lines in any combination to construct boxes around fields or groups of fields.

PARTN
Specifies the default partition to be associated with maps in this mapset. If the ACTIVATE option is specified, the specified partition is also activated when maps in this mapset are output to a terminal that supports partitions. This option is overridden by the PARTN operand of the DFHMDI macro, which is in turn overridden by any OUTPARTN or ACTPARTN option on the SEND MAP command, or the INPARTN option on a RECEIVE MAP command.

The PARTN operand is ignored if the target terminal does not support partitions, or if there is no partition set associated with the transaction.

PS
Specifies that programmed symbols are to be used. This is overridden by the PS operand of the DFHMDI macro, which is in turn overridden by the PS operand of the DFHMDF macro.

BASE Specifies that the base symbol set is to be used.

psid Specifies a single EBCDIC character, or a hexadecimal code of the form X'nn', that identifies the set of programmed symbols to be used.

The PS operand is ignored unless the terminal supports programmed symbols, as indicated by PROGSYMBOLS(YES) on the RDO TYPETERM definition.

SOSI
Indicates that the field can contain a mixture of EBCDIC and DBCS data. The DBCS subfields in an EBCDIC field are delimited by SO (shift out) and SI (shift in) characters. SO and SI both occupy a single screen position (normally displayed as a blank). They can be included in any non-DBCS field on output provided they are correctly paired. The terminal user can transmit them inbound if they are already present in the field, but can add them to an EBCDIC field only if the field has the SOSI attribute.

STORAGE
The meaning of this operand depends upon the language in which application programs are written, as follows:

For a **COBOL** program, STORAGE=AUTO specifies that the symbolic description maps in the mapset are to occupy separate (that is, not redefined) areas of storage. This operand is used when the symbolic description maps are copied into the working-storage section and the storage for the separate maps in the mapset is to be used concurrently.

For a **C** program, STORAGE=AUTO specifies that the symbolic description maps are to be defined as having the automatic storage class. If STORAGE=AUTO is not specified, they are declared as pointers.

For a **PL/I** program, STORAGE=AUTO specifies that the symbolic description maps are to be declared as having the AUTOMATIC storage class. If STORAGE=AUTO is not specified, they are declared as BASED.

For an **assembler language** program, STORAGE=AUTO specifies that individual maps in a mapset are to occupy separate areas of storage instead of overlaying one another.

For all languages:

- You cannot specify both BASE=name and STORAGE=AUTO for the same mapset.
- If STORAGE=AUTO is specified and TIOAPFX is not, TIOAPFX=YES is assumed.

SUFFIX
Specifies a one-character, user-defined, device-dependent suffix for this mapset, as an alternative to a suffix generated by the TERM operand. The suffix specified by this operand should match the value of a transaction defined on the ALTSUFFIX attribute of a TYPETERM definition, or ALTSFX in the terminal control table TYPE=TERMINAL. Use a numeric value to avoid conflict with suffixes generated by the TERM operand.

TERM
Specifies the type of terminal or logical unit (LU) associated with the mapset. If no terminal type or LU is specified, 3270 is assumed. The terminal types and LUs you can specify, together with their generated suffixes, are shown in Table 26.

In addition, you should note the following:

If ALL is specified, ensure that device-dependent characters are not included in the mapset and that format characteristics such as page size are suitable for all input/output operations (and all terminals) in which the mapset is applied. For example, some terminals are limited to 480 bytes, others to 1920 bytes; the 3604 is limited to six lines of 40 characters each. Within these guidelines, use of ALL can offer important advantages. Because an assembly run is required for each map generation, the use of ALL, indicating that one map is to be used for more than one terminal, can result in significant time and storage savings.

However, better run-time performance for maps used by single terminal types is achieved if the terminal type (rather than ALL) is specified. Alternatively, BMS support for device-dependent mapsets can be bypassed by specifying NODDS in the BMS operand of the system initialization parameters.

Table 26. BMS terminal types

Type	Suffix	Notes
CRLP	A	Card-reader-in/line-printer-out
TAPE	B	
DISK	C	
TWX	D	
1050	E	
2740	F	
2741	G	
2770	I	
2780	J	
3780	K	
3270-1 (40-column)	L	
3270-2 (80-column)	M	
INTLU/3767/3770I/SCS	p	All interactive LUs including 3790 full-function LU and SCS printer LUs (3270 and 3790).
2980	Q	
2980-4	R	
3270	blank	Default if TERM omitted. Same as ALL; used when no need to distinguish between models.
3601	U	
3653	V	Plus host-conv (3653) LU.
3650UP	W	Plus interpreter LU.

Table 26. BMS terminal types (continued)

Type	Suffix	Notes
3650/3270	X	Plus host-conv (32700 LU.
BCHLU/3770B	Y	Plus all batch and BDI LUs.
ALL (all of the above)	blank	

TIOAPFX
Specifies whether BMS should include a filler in the symbolic description maps to allow for the unused TIOA prefix.

YES Specifies that the filler should be included in the symbolic description maps. If TIOAPFX=YES is specified, all maps in the mapset have the filler, except when TIOAPFX=NO is specified on the DFHMDI macro. TIOAPFX=YES is the default if STORAGE=AUTO is specified. TIOAPFX=YES should **always** be used for command-level application programs.

NO Is the default, unless STORAGE=AUTO is specified, and specifies that the filler is not to be included. The filler can still be included for a map if TIOAPFX=YES is specified on DFHMDI.

TRANSP
Determines whether the background of an alphanumeric field is transparent or opaque, that is, whether an underlying (graphic) presentation space is visible between the characters.

TRIGRAPH
Specifies trigraph sequences to be used in C language symbolic description maps.

When TRIGRAPH=YES, trigraph sequences are produced:

{ prints as ??<

} prints as ??>

[prints as ??(

] prints as ??)

This option is available only for programs written in C.

TYPE
Specifies the type of map to be generated using the definition. Both types of map must be generated before the mapset can be used by an application program. If aligned symbolic description maps are required, you should ensure that you specify SYSPARM=ADSECT and SYSPARM=AMAP when you assemble the symbolic and physical maps respectively.

DSECT
Specifies that a symbolic description map is to be generated. Symbolic description maps must be copied into the source program before it is translated and compiled.

MAP Specifies that a physical map is to be generated. Physical maps must be assembled or compiled, link-edited, and cataloged in the CICS program library before an application program can use them.

If both map and DSECT are to be generated in the same job, the SYSPARM option can be used in the assembler job execution step.

VALIDN

Specifies that:

- Validation is to be used on an 8775 terminal
- This field can be processed by the BMS global user exits

This is overridden by the VALIDN operand of the DFHMDI macro, which is in turn overridden by the VALIDN of the DFHMDF macro.

MUSTFILL

Specifies that the field must be filled completely with data. An attempt to move the cursor from the field before it has been filled, or to transmit data from an incomplete field, raises the INHIBIT INPUT condition.

MUSTENTER

Specifies that data must be entered into the field, though need not fill it. An attempt to move the cursor from an empty field raises the INHIBIT INPUT condition.

TRIGGER

Specifies that this field is a trigger field. Trigger fields are discussed in .

USEREXIT

Specifies that this field is to be processed by the BMS global user exits, XBMIN and XBMOUT, if this field is received or transmitted in a 3270 datastream when the respective exit is enabled.

The MUSTFILL, MUSTENTER, and TRIGGER specifications are valid only for terminals that support the field validation extended attribute, otherwise they are ignored. The USEREXIT specification applies to all 3270 devices.

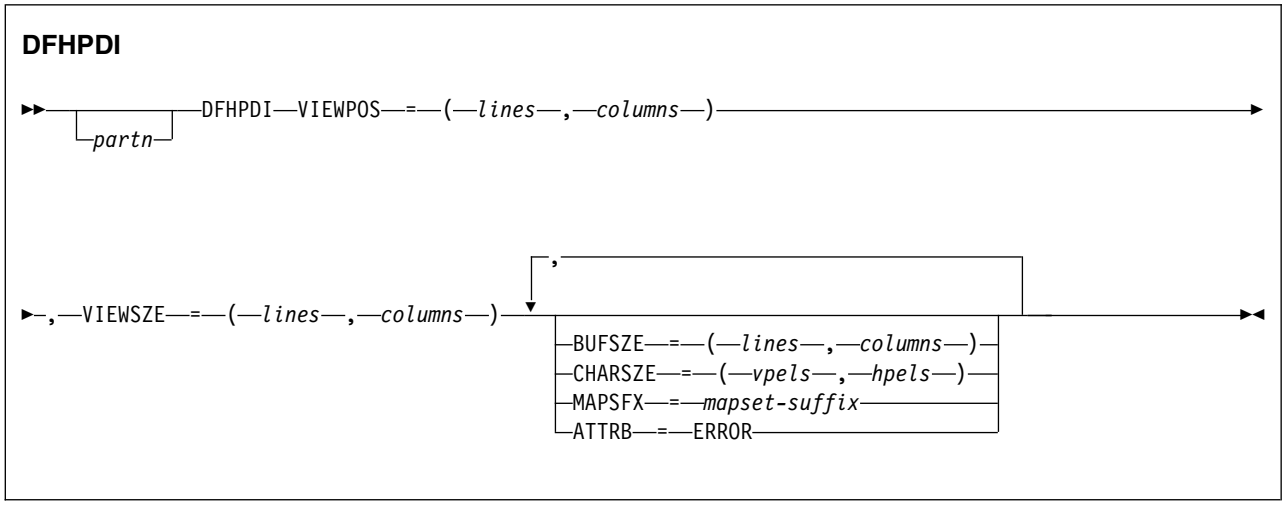
Note: The USEREXIT specification is unconnected with the field validation extended attribute as defined in the 3270 datastream architecture.

VTAB

Specifies one or more tab positions for use with interactive and batch logical units and SCS printers having vertical forms control.

DFHPDI

A partition set contains one or more partitions. Each partition is defined by coding a partition definition macro.



"partn" is a partition name (1–2 characters). It allows you to refer to the partition in your application programs.

Every partition in a partition set must have a different name. Only the error partition can be unnamed (see ATTRB=ERROR operand).

Partitions are defined by coding the macros DFHPSD (partition set definition) and DFHPDI (partition definition). Each partition definition must be part of a partition set definition.

Operands

ATTRB
specifies that error messages are to be directed to this partition whenever possible. The partition is cleared before an error message is displayed. The RDO TYPETERM option ERRHILIGHT is honored, but the LASTLINE option is ignored.

BUFSZE(lines,columns)
specifies the size of the presentation space for the partition. Device limitations mean that the "columns" value must be equal to the "columns" value specified by the VIEWSZE operand. The "lines" value can be greater than or, by default, equal to the value specified by the VIEWSZE operand. A greater lines value implies that the target terminal supports vertical scrolling.

CHARSZE(vpels,hpels)
specifies the size of the character cell to be reserved for each character displayed in a partition. You specify the size as numbers of vertical picture elements (vpels) and numbers of horizontal picture elements (hpels). You can specify this operand on either the DFHPSD macro only, or on both the DFHPSD and DFHPDI macros. The values specified in the DFHPSD become the defaults for all partitions in the partition set. You can override these defaults for individual partitions by coding CHARSZE in the DFHPDI macro.

MAPSFX(mapset-suffix)
specifies the partition's 1-character mapset suffix. BMS uses the suffix to select mapset versions in the same way as for the RDO option ALTSUFFIX. If this operand is omitted, a suffix L is assumed if the “columns” value of the BUFSZE operand is less than or equal to 40; otherwise M is assumed.

VIEWPOS(lines,columns)
specifies the position of the top left-hand corner of this partition's viewport. You specify the position in numbers of lines and numbers of columns.

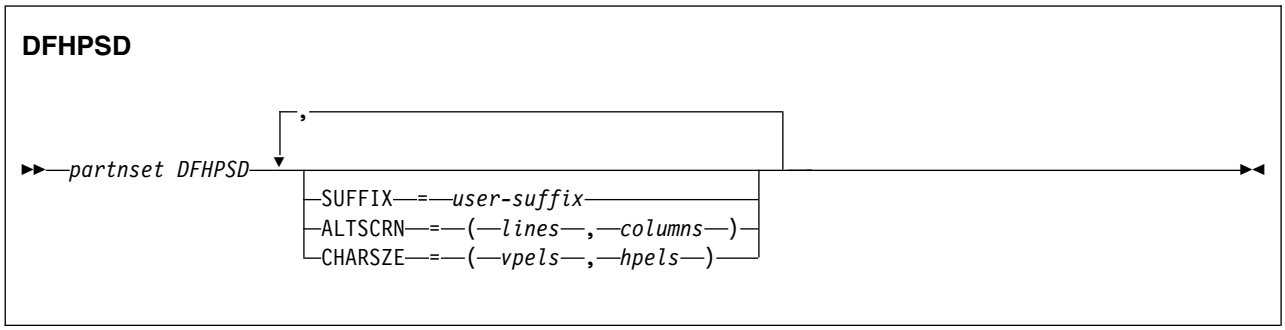
The DFHPDI macro checks that viewports do not overlap. If you have coded the RDO TYPETERM ALTSCREEN option, or the ALTSCRN operand of the DFHPSD macro, DFHPDI also checks that all viewports fit within the usable area of the terminal screen.

Note: The information given here on positioning viewports is necessarily brief. For more information you should consult the component description for the device you are using.

VIEWSZE(lines,columns)
specifies the size, in lines and columns, of the partition's viewport. The DFHPDI macro checks that viewports do not overlap. If you code the RDO TYPETERM ALTSCREEN option, or the ALTSCRN operand of the DFHPSD macro partition set definition macro, DFHPDI checks that the partitions all fit within the usable area of the display screen.

DFHPSD

Each partition set definition contains a single DFHPSD macro followed by one or more DFHPDI macros, and ending with a DFHPSD TYPE=FINAL partition set definition macro.



“partnset” is a partition set name (1–6 characters).

Partitions are defined by coding the macros DFHPSD (partition set definition) and DFHPDI (partition definition). Each partition definition must be part of a partition set definition.

Operands

ALTSCRN(lines,columns)
specifies the size, in characters, of the usable area of the target terminal. This is normally the same as the RDO TYPETERM option ALTSCREEN. You use ALTSCRN to ensure that the viewports of partitions within a partition set fit into the usable area of the screen.

CHARSZE(vpels,hpels)
specifies the size of the character cell to be reserved for each character displayed in a partition. You specify the size as numbers of vertical picture elements (vpels) and numbers of horizontal picture elements (hpels). You can specify this operand on either the DFHPSD macro only, or on both the DFHPSD and DFHPDI macros. The values specified in this operand become the defaults for all partitions in the partition set. You can override this default for individual partitions by coding CHARSZE in the DFHPDI macro.

SUFFIX(user-suffix)
specifies a 1-character user suffix for this version of the partition set. It allows different versions of a partition set to be associated with different terminals. When the partition set is to be loaded, CICS looks for a version whose suffix matches the RDO TYPETERM option ALTSUFFIX. If it cannot find the correct partition set version, it loads a version with a default suffix (M or L). If it cannot find a suffixed version either, it loads an unsuffixed one. If it cannot find this, it abends with APCT.

Ending DFHPSD

```
[partnset] DFHPSD TYPE=FINAL
```

The PARTNSET name (if specified) must match that specified on the DFHPSD macro that started the partition set definition.

Notices

This information was developed for products and services offered in the U.S.A. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 rights 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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

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

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

© Copyright IBM Corp. 1989, 2017

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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

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

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

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

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

Programming interface information

CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- RACF security overview
- Developing for external interfaces
- Reference: application developmenth
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- Reference: diagnostics

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services
- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- Supplied Transactions
- CICSPlex SM Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

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.

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

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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

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

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below:

For the CICSPlex® SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user’s user name and other personally identifiable information for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other personally identifiable information for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect no personally identifiable information. These cookies cannot be disabled.

For CICS Explorer®:
Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user’s user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled “Cookies, Web Beacons and Other Technologies” and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index

Special characters

> 32K COMMAREAs (channels)
  ASSIGN command 47
  CHANNEL option of RETURN command 459
  CHANNEL option of XCTL command 747
  DELETE CHANNEL command 129
  DELETE CONTAINER (CHANNEL) command 132
  GET CONTAINER (CHANNEL) command 234
  MOVE CONTAINER (CHANNEL) command 345
  PUT CONTAINER (CHANNEL) command 357
  QUERY CHANNEL command 365
  START CHANNEL command 576

Numerics

2260 Display Station
  CONVERSE command 97
  RECEIVE command 423
  SEND command 495
2265 Display Station
  CONVERSE command 97
  RECEIVE command 423
  SEND command 495
2980 general banking terminal system
  RECEIVE/SEND commands 495
2980 General Banking Terminal System
  DFH2980 structure 425
  output control 425
  output to common buffer 425
  passbook control 424
  RECEIVE/SEND commands 424
3270 information display system
  logical unit 484
3270 Information Display System 317
  logical unit 86, 293, 415
3600 finance communication system
  3614 logical unit 486
3600 Finance Communication System
  3601 logical unit 86, 87, 416, 485
  3614 logical unit 87, 416
  pipeline logical units 484
3630 Plant Communication System
  RECEIVE command 416
  SEND command 485
3650 Host Command Processor
  CONVERSE command 89
3650 Logical Units
  RECEIVE command 417
3650 Store System
  host conversational
    LU 3270 487
    LU 3653 487
  interpreter logical unit 88, 300, 305, 417, 486
3650/3680 Full-function Logical Unit
  RECEIVE command 418
  SEND command 489
3650/3680 Store System
  host command processor LU 488
3680 Host Command Processor
  CONVERSE command 89
3680 Programmable Store System
  host command processor LU 488
3740 Data Entry System 298, 299
3767 communication terminal
  interactive logical unit 488
3767 Communication Terminal
  interactive logical unit 90, 417
3770 data communication system
  batch logical unit 488
3770 Data Communication System
  batch logical unit 90, 418
3770 Full Function Logical Unit
  RECEIVE command 418
  SEND command 489
3770 interactive logical unit
  RECEIVE command 418
3770 Interactive Logical Unit
  SEND command 488
3790 communication system
  3270-display logical unit 490
  full-function logical unit 489
  SCS printer logical unit 489
3790 Communication System
  3270-display logical unit 91, 426
  Full Function Logical Unit 91
  full-function logical unit 418

A

ABCODE option
  ABEND command 25
  ASSIGN command 43
  CHECK ACQPROCESS command 75
  CHECK ACTIVITY command 77
  FETCH CHILD command 191, 192, 193
  INQUIRE ACTIVITYID command 269
ABDUMP option
  ASSIGN command 43
ABEND command 25
ABEND exit, reactivating 262
abend support commands 17
abnormal termination, task 262
ABOFFSET option
  ASSIGN command 43
ABPROGRAM option
  ASSIGN command 43
  CHECK ACQPROCESS command 75
  CHECK ACTIVITY command 77
  INQUIRE ACTIVITYID command 269
absolute expressions 7
ABSTIME option
  ASKTIME command 40
  CONVERTTIME command 103
  FORMATTIME command 197
  INQUIRE TIMER command 277
access to system information
  ADDRESS command 31
  ADDRESS SET command 33
  ASSIGN command 41
  CICS storage areas 31, 33
ACCUM option
  SEND CONTROL command 500
  SEND MAP command 505
  SEND TEXT command 519
ACEE option
  ADDRESS command 31
ACQACTIVITY option
  CANCEL (BTS) command 68
  CHECK ACTIVITY command 77
  DELETE CONTAINER (BTS) command 130
  FORCE TIMER command 194
  GET CONTAINER (BTS) command 232
  LINK ACTIVITY command 336
  PUT CONTAINER (BTS) command 355
  RESUME command 452
  RUN command 477
  SUSPEND (BTS) command 594
ACQPROCESS option
  CANCEL (BTS) command 68
  CHECK ACQPROCESS command 75
  DELETE CONTAINER (BTS) command 131
  FORCE TIMER command 195
  GET CONTAINER (BTS) command 232
  LINK ACQPROCESS command 333
  PUT CONTAINER (BTS) command 355
  RESET ACQPROCESS command 445
  RESUME command 452
  RUN command 477
  SUSPEND (BTS) command 594
ACQUIRE command 27
ACTION option
  WEB CONVERSE command 639
  WEB SEND command (Client) 699
  WEB SEND command (Server) 691
  WRITE OPERATOR command 727
ACTIVE mode, of an activity 270
activities
  destruction of 128
  modes 270
  processing states 270
ACTIVITY option
  ASSIGN command 43
  CANCEL (BTS) command 69
  CHECK ACTIVITY command 77
  DEFINE ACTIVITY command 104

© Copyright IBM Corp. 1989, 2017

ACTIVITY option *(continued)*
DELETE ACTIVITY command 128
DELETE CONTAINER (BTS) command 131
GET CONTAINER (BTS) command 232
GETNEXT ACTIVITY command 252
INQUIRE ACTIVITYID command 270
LINK ACTIVITY command 336
PUT CONTAINER (BTS) command 355
RESET ACTIVITY command 446
RESUME command 452
RUN command 477
SUSPEND (BTS) command 594
activity-related commands
ACQUIRE 27
CANCEL (BTS) 68
CHECK ACQPROCESS 74
CHECK ACTIVITY 76
DEFINE ACTIVITY 104
DEFINE PROCESS 112
DELETE ACTIVITY 128
INQUIRE ACTIVITYID 269
INQUIRE PROCESS 275
LINK ACQPROCESS 332
LINK ACTIVITY 335
RESET ACQPROCESS 444
RESET ACTIVITY 446
RESUME 451
RUN 475
STARTBROWSE ACTIVITY 587
SUSPEND (BTS) 593
ACTIVITYBUSY condition
ACQUIRE command 28
CANCEL (BTS) command 69
CHECK ACTIVITY command 78
DELETE ACTIVITY command 128
LINK ACTIVITY command 336
RESET ACTIVITY command 446
RESUME command 452
RUN command 477
SUSPEND (BTS) command 594
ACTIVITYERR condition
ACQUIRE command 28
CANCEL (BTS) command 69
CHECK ACTIVITY command 78
DEFINE ACTIVITY command 105
DELETE ACTIVITY command 128
DELETE CONTAINER (BTS) command 131
GET CONTAINER (BTS) command 233
GETNEXT ACTIVITY command 253
INQUIRE ACTIVITYID command 271
INQUIRE CONTAINER command 273
INQUIRE EVENT command 275
INQUIRE TIMER command 277
LINK ACTIVITY command 337
MOVE CONTAINER (BTS) command 345
PUT CONTAINER (BTS) command 356
RESET ACTIVITY command 446
ACTIVITYERR condition *(continued)*
RESUME command 452
RUN command 478
STARTBROWSE ACTIVITY command 588
STARTBROWSE CONTAINER command 589
STARTBROWSE EVENT command 591
SUSPEND (BTS) command 594
ACTIVITYID option
ACQUIRE command 28
ASSIGN command 44
DEFINE ACTIVITY command 104
GETNEXT ACTIVITY command 252
GETNEXT PROCESS command 257
INQUIRE ACTIVITYID command 270
INQUIRE CONTAINER command 272
INQUIRE EVENT command 274
INQUIRE PROCESS command 276
INQUIRE TIMER command 277
STARTBROWSE ACTIVITY command 587
STARTBROWSE CONTAINER command 589
STARTBROWSE EVENT command 591
ACTPARTN option
SEND CONTROL command 500
SEND MAP command 505
SEND TEXT command 519
ADD SUBEVENT command 29
ADDRESS command 31
ADDRESS SET command 33
address, cursor 846
ADS descriptor 855
ADS value
DFHMSD 877
ADSL value
DFHMSD 877
ADW1 25
AFTER option
DEFINE TIMER command 115
POST command 351
ROUTE command 471
START command 567
AID option
RECEIVE MAP MAPPINGDEV command 434
ALARM option
SEND CONTROL command 500
SEND MAP command 505
SEND MAP MAPPINGDEV command 512
SEND TEXT command 519
SEND TEXT NOEDIT command 527
ALARM value
DFHMDI 868
DFHMSD 877
ALIGNED attribute
PL/I 7
ALL option
SEND PAGE command 515
ALLOCATE (APPC) command 33
ALLOCATE (LUTYPE6.1) command 37
ALLOCATE (MRO) command 38
ALLOCERR condition
SPOOLCLOSE command 548
SPOOLOPEN INPUT command 550
SPOOLOPEN OUTPUT command 555
SPOOLREAD command 557
SPOOLWRITE command 560
ALTER option
QUERY SECURITY command 369
ALTERNATE option
CONVERSE (non-z/OS Communications Server) command 98
CONVERSE (z/OS Communications Server) command 92
SEND (non-z/OS Communications Server) command 496
SEND (z/OS Communications Server) command 490
SEND CONTROL command 500
SEND MAP command 505
SEND TEXT command 519
SEND TEXT NOEDIT command 528
ALTSCRN operand
DFHPSD 889
ALTSCRNHT option
ASSIGN command 44
ALTSCRNWD option
ASSIGN command 44
AND option
DEFINE COMPOSITE EVENT command 107
ANY option
FETCH CHILD command 191
ANYKEY option
HANDLE AID command 264
APLKYBD option
ASSIGN command 44
APLTEXT option
ASSIGN command 44
APPC basic conversations commands 17
APPC logical unit
acquiring session to 33
initiating conversation with 80
returning mapped sessions to CICS 201
sending and receiving 83
APPC mapped conversations
abending 287
commands 17
ensuring transmission of accumulated data 623
extracting attributes of 174
informing partner of error 304
issuing a positive response 292
receiving data 413
requesting change of direction 319
retrieving values from attach header 180
returning sessions to CICS 201
sending data 481
APPEND option
PUT CONTAINER (CHANNEL) command 357
PUT64 CONTAINER command 361

APPLICATION option 44
INVOKE APPLICATION command 279
application performance, monitoring 341
APPLID option
ASSIGN command 44
APPNOTFOUND condition
INVOKE APPLICATION command 280
argument values
assembler language 7
C 4
COBOL 3
PL/I 6
AS option
MOVE CONTAINER (BTS) command 344
MOVE CONTAINER (CHANNEL) command 346
ASA option
SPOOLOPEN OUTPUT command 553
ASIS option
CONVERSE (non-z/OS Communications Server) command 98
CONVERSE (z/OS Communications Server) command 92
RECEIVE (non-z/OS Communications Server) command 426
RECEIVE (z/OS Communications Server) command 419
RECEIVE MAP command 431
RECEIVE PARTN command 436
SEND (non-z/OS Communications Server) command 496
ASKIP value
DFHMDF 857
ASKTIME command 40
ASRAINTRPT option
ASSIGN command 44
ASRAKEY option
ASSIGN command 44
ASRAPSW option
ASSIGN command 45
ASRAPSW16 option
ASSIGN command 45
ASRAREGS option
ASSIGN command 45
ASRAREGS64 option
ASSIGN command 45
ASRASPC option
ASSIGN command 45
ASRASTG option
ASSIGN command 46
assembler language
argument values 7
LENGTH option default 8
register contents 13
translated code 13
assembly language
program exit 13
ASSIGN command 41, 44, 50, 51, 52
Asynchronous API commands 17
asynchronous interrupt 841
ASYNCHRONOUS option
RUN command 477
AT option
DEFINE TIMER command 115
DOCUMENT INSERT command 146
POST command 351
ROUTE command 471
START command 567
attach
start a task 572
ATTACHID option
BUILD ATTACH (LUTYPE6.1) command 61
BUILD ATTACH (MRO) command 64
CONVERSE (non-z/OS Communications Server) command 98
CONVERSE (z/OS Communications Server) command 92
EXTRACT ATTACH (LUTYPE6.1) command 168
EXTRACT ATTACH (MRO) command 171
SEND (non-z/OS Communications Server) command 496
SEND (z/OS Communications Server) command 491
attention identifier (AID) 264
ATTRB operand
DFHMDF 857
DFHPDI 888
attributes
control character list, DFHBMSCA 849
AUTHENTICATE option
EXTRACT TCPIP command 182
WEB CONVERSE command (Client) 640
WEB SEND command (Client) 700
authentication commands 18
AUTOPAGE option
SEND PAGE command 515
AUXILIARY option
WRITEQ TS command 731

B

back out to a syncpoint 596
BASE operand
DFHMSD 877
BASE value
DFHMDF 857
DFHMDI 868
DFHMSD 877
basic mapping support (BMS)
ADS descriptor 855
commands 18
completing a logical message 514
deleting a logical message 353
determining input partition 435
field definition macro 855, 857
full BMS
RECEIVE MAP command 430
RECEIVE PARTN command 435
SEND CONTROL command 499
SEND MAP command 504
basic mapping support (BMS) *(continued)*
full BMS *(continued)*
SEND PAGE 514
SEND PARTNSET 517
SEND TEXT command 518
SEND TEXT MAPPED 525
SEND TEXT NOEDIT 527
full-function BMS
PURGE MESSAGE 353
map definition macro 855, 868
mapping input data 430
mapping input data with MAPPINGDEV 433
mapset definition macro 855, 877
minimum BMS
RECEIVE MAP command 430
RECEIVE MAP MAPPINGDEV command 433
SEND CONTROL command 499
SEND MAP command 504
SEND MAP MAPPINGDEV command 511
partition definition macro 856, 888
partition set definition macro 856, 889
related constants 849
routing a logical message 471
sending previously mapped data 525
sending user-defined data stream 527
standard BMS
RECEIVE MAP command 430
RECEIVE PARTN command 435
SEND CONTROL command 499
SEND MAP command 504
SEND PARTNSET 517
SEND TEXT command 518
batch data interchange (BDI)
add record to data set 290
commands 18
conditions 307
delete a record from data set 301
read record from data set 313
request next record number 306
send data to output device 317
terminate data set 288, 297
update a record in data set 315
wait for function completion 322
batch logical unit, 3770 90, 418, 488
BELOW option
GETMAIN command 246
BIF DEEDIT command 59
BIF DIGEST command 60
big COMMAREAs (channels)
ASSIGN command 47
DELETE CHANNEL command 129
DELETE CONTAINER (CHANNEL) command 132
QUERY CHANNEL command 365
big COMMAREAs, channels 129, 132, 234, 345, 357, 365, 459, 576, 747
BINARY option
DOCUMENT INSERT command 146
BLANK value
DFHMDF 857
BLINK value
DFHMDF 857

BLINK value *(continued)*
DFHMDI 868
DFHMSD 877
BLOCK value
DFHMDI 868
DFHMSD 877
BODYCHARSET option
WEB CONVERSE command 644
WEB RECEIVE command (Client) 683
WEB RECEIVE command (Server) 676
BOOKMARK option
DOCUMENT INSERT command 146
BOTTOM value
DFHMDI 868
BRDATA option
START BREXIT command 573
BRDATALENGTH option
START BREXIT command 573
BREXIT option
START BREXIT command 573
bridge (3270)
start a task 573
BRIDGE option
ASSIGN command 46
browse operation
ending 158
read next record 385
read previous record 395
reset starting point 447
starting 580
BROWSETOKEN option
ENDBROWSE ACTIVITY command 160
ENDBROWSE CONTAINER (CHANNEL) command 162
ENDBROWSE CONTAINER command 161
ENDBROWSE EVENT command 162
ENDBROWSE PROCESS command 163
GETNEXT ACTIVITY command 252
GETNEXT CONTAINER (CHANNEL) command 255
GETNEXT CONTAINER command 254
GETNEXT EVENT command 255
GETNEXT PROCESS command 257
STARTBROWSE ACTIVITY command 588
STARTBROWSE CONTAINER command 589, 590
STARTBROWSE EVENT command 591
STARTBROWSE PROCESS command 592
browsing commands
ENDBROWSE ACTIVITY 160
ENDBROWSE CONTAINER 161
ENDBROWSE CONTAINER (CHANNEL) 161
ENDBROWSE PROCESS 163
GETNEXT ACTIVITY 252
GETNEXT CONTAINER 253
GETNEXT CONTAINER (CHANNEL) 254
browsing commands *(continued)*
GETNEXT EVENT 255
GETNEXT PROCESS 257
INQUIRE ACTIVITYID 269
INQUIRE CONTAINER 272
INQUIRE EVENT 274
INQUIRE PROCESS 275
INQUIRE TIMER 276
STARTBROWSE ACTIVITY 587
STARTBROWSE CONTAINER 588
STARTBROWSE CONTAINER (CHANNEL) 590
STARTBROWSE PROCESS 592
BRT value
DFHMDF 857
BTRANS option
ASSIGN command 46
BTS commands
ACQUIRE 27
ADD SUBEVENT 29
CANCEL (BTS) 68
CHECK ACQPROCESS 74
CHECK ACTIVITY 76
CHECK TIMER 79
DEFINE ACTIVITY 104
DEFINE COMPOSITE EVENT 106
DEFINE INPUT EVENT 111
DEFINE PROCESS 112
DEFINE TIMER 115
DELETE ACTIVITY 128
DELETE CONTAINER (BTS) 130
DELETE EVENT 135
DELETE TIMER 136
ENDBROWSE ACTIVITY 160
ENDBROWSE CONTAINER 161
ENDBROWSE EVENT 162
ENDBROWSE PROCESS 163
FORCE TIMER 194
GET CONTAINER (BTS) 232
GETNEXT ACTIVITY 252
GETNEXT CONTAINER 253
GETNEXT EVENT 255
GETNEXT PROCESS 257
INQUIRE ACTIVITYID 269
INQUIRE CONTAINER 272
INQUIRE EVENT 274
INQUIRE PROCESS 275
INQUIRE TIMER 276
LINK ACQPROCESS 332
LINK ACTIVITY 335
MOVE CONTAINER (BTS) 343
PUT CONTAINER (BTS) 354
REMOVE SUBEVENT 439
RESET ACQPROCESS 444
RESET ACTIVITY 446
RESUME 451
RETRIEVE REATTACH EVENT 456
RETRIEVE SUBEVENT 458
RUN 475
STARTBROWSE ACTIVITY 587
STARTBROWSE CONTAINER 588
STARTBROWSE EVENT 591
STARTBROWSE PROCESS 592
SUSPEND (BTS) 593
TEST EVENT 597
BUFFER option
GDS RECEIVE command 226
BUFFER option *(continued)*
RECEIVE (non-z/OS Communications Server) command 427
RECEIVE (z/OS Communications Server) command 419
BUFSZE operand
DFHPDI 888
BUILD ATTACH (LUTYPE6.1) command 61
BUILD ATTACH (MRO) command 64
built-in functions
commands 18
BUSY condition
DELETE COUNTER command 133
BYTEOFFSET option
GET CONTAINER (CHANNEL) command 235
GET64 CONTAINER command 258

C

C language
ADDRESS COMMAREA 32
ADDRESS EIB 32
argument values 4
LENGTH option default 5
translated code 12
CADDRLENGTH option
EXTRACT TCPIP command 182
CANCEL (BTS) command 68
CANCEL command 66
CANCEL option
ABEND command 25
HANDLE ABEND command 263
CANCELLING mode, of an activity 270
CARD option
ISSUE ABORT command 289
ISSUE END command 297
ISSUE SEND command 318
ISSUE WAIT command 322
CASE operand
DFHMDF 857
CBIDERR condition
ALLOCATE (APPC) command 36
ALLOCATE (LUTYPE6.1) command 38
CONVERSE (non-z/OS Communications Server) command 101
CONVERSE (z/OS Communications Server) command 95
EXTRACT ATTACH (LUTYPE6.1) command 171
EXTRACT ATTACH (MRO) command 174
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 493
CBUFF option
SEND (non-z/OS Communications Server) command 496
CCSID option
GET CONTAINER (CHANNEL) command 235
GET64 CONTAINER command 258

CCSIDERR condition
GET CONTAINER (CHANNEL) command 238
GET64 CONTAINER command 261
PUT CONTAINER (CHANNEL) command 359
PUT64 CONTAINER command 363
SOAPFAULT CREATE command 546
WSACONTEXT BUILD command 738
CEE3250C 25
CEEMSG 25
CERTIFICATE option
EXTRACT CERTIFICATE command 177
WEB OPEN command 658
CHANGE PASSWORD command 72
CHANGE PHRASE command 70
CHANGE TASK command 74
CHANGED condition
DELETE command 120
REWRITE command 467
CHANGETIME option
VERIFY PASSWORD command 614
VERIFY PHRASE command 617
channel commands
CHANNEL option of RETURN command 459
CHANNEL option of XCTL command 747
DELETE CHANNEL 129
DELETE CONTAINER (CHANNEL) 132
ENDBROWSE CONTAINER (CHANNEL) 161
GET CONTAINER (CHANNEL) 234
GET64 CONTAINER 258
GETNEXT CONTAINER (CHANNEL) 254
MOVE CONTAINER (CHANNEL) 345
PUT CONTAINER (CHANNEL) 357
PUT64 CONTAINER 361
QUERY CHANNEL 365
START CHANNEL 576
STARTBROWSE CONTAINER (CHANNEL) 590
Channel commands 20
CHANNEL option
ASSIGN command 47
DELETE CHANNEL command 129
DELETE CONTAINER (CHANNEL) command 132
FETCH CHILD command 191, 192, 193
GET CONTAINER (CHANNEL) command 235
GET64 CONTAINER command 259
INVOKE APPLICATION command 279
LINK command 326
MOVE CONTAINER (CHANNEL) command 347
PUT CONTAINER (CHANNEL) command 357
PUT64 CONTAINER command 361
QUERY CHANNEL command 365
CHANNEL option *(continued)*
RETURN command 460
RUN TRANSID command 479
START TRANSID (CHANNEL) command 576
TRANSFORM XMLTODATA command 604
WEB CONVERSE command 640
WEB SEND command (Client) 700
WEB SEND command (Server) 691
XCTL command 747
CHANNELERR condition
DELETE CHANNEL command 130
DELETE CONTAINER (CHANNEL) command 132
GET CONTAINER (CHANNEL) command 238
GET64 CONTAINER command 261
INVOKE APPLICATION command 281
LINK command 329
MOVE CONTAINER (CHANNEL) command 347
PUT CONTAINER (CHANNEL) command 359
PUT64 CONTAINER command 363
QUERY CHANNEL command 365
RETURN command 462
SIGNAL EVENT command 531
SOAPFAULT ADD command 542
SOAPFAULT CREATE command 546
SOAPFAULT DELETE command 547
START TRANSID (CHANNEL) command 576
WEB CONVERSE command 648
WEB RECEIVE command (Client) 687
WEB RECEIVE command (Server) 681
WEB SEND command (Client) 706
WEB SEND command (Server) 696
WSACONTEXT BUILD command 738
XCTL command 749
channels
ASSIGN command 47
channels as large COMMAREAs 129, 132, 234, 345, 357, 365, 459, 576, 747
CHARACTERSET option
DOCUMENT RETRIEVE command 150
WEB CONVERSE command 640
WEB READ FORMFIELD command 665
WEB RECEIVE command (Server) 676
WEB SEND command (Client) 700
WEB SEND command (Server) 691
WEB STARTBROWSE FORMFIELD command 708
CHARSZE operand
DFHPDI 888
DFHPSD 889
CHECK ACQPROCESS command 74
CHECK ACTIVITY command 76
CHECK TIMER command 79
CHILD option
FETCH CHILD command 191, 192, 193
RUN TRANSID command 480
CHUNKING option
WEB SEND command (Client) 700
WEB SEND command (Server) 691
CICS business transaction services (BTS) commands 18
CICS Event processing commands
SIGNAL EVENT 530
CICS Web Interface (CWI) commands
DOCUMENT CREATE 141
DOCUMENT DELETE 145
DOCUMENT INSERT 146
DOCUMENT RETRIEVE 149
DOCUMENT SET 151
EXTRACT CERTIFICATE 177
CICS Web support commands
CONVERSE WEB 637
EXTRACT WEB 186
WEB CLOSE 635
WEB CONVERSE 637
WEB ENDBROWSE FORMFIELD 650
WEB ENDBROWSE HTTPHEADER 651
WEB ENDBROWSE QUERYPARM 651
WEB EXTRACT 652
WEB OPEN 657
WEB PARSE URL 662
WEB READ FORMFIELD 664
WEB READ HTTPHEADER 667
WEB READ QUERYPARM 668
WEB READNEXT FORMFIELD 670
WEB READNEXT HTTPHEADER 671
WEB READNEXT QUERYPARM 673
WEB RECEIVE 674
WEB RECEIVE (Client) 681
WEB RETRIEVE 688
WEB SEND (Client) 698
WEB SEND (Server) 689
WEB STARTBROWSE FORMFIELD 708
WEB STARTBROWSE HTTPHEADER 710
WEB STARTBROWSE QUERYPARM 711
WEB WRITE HTTPHEADER 712
CICSDATAKEY option
GETMAIN command 246
GETMAIN64 command 250
CIPHERS option
WEB OPEN command 658
CLASS option
SPOOLOPEN INPUT command 550
SPOOLOPEN OUTPUT command 553
CLEAR option
HANDLE AID command 264
client requests
extracting information 177
CLIENTADDR option
EXTRACT TCPIP command 183

CLIENTADDRNU option
EXTRACT TCPIP command 183
CLIENTCONV option
WEB CONVERSE command 646
WEB RECEIVE command (Client) 683
WEB SEND command (Client) 702
CLIENTNAME option
EXTRACT TCPIP command 183
CLNTADDR6NU option
EXTRACT TCPIP command 183
CLNTCODEPAGE option
DOCUMENT RETRIEVE command 150
WEB READ FORMFIELD command 665
WEB RECEIVE command (Server) 677
WEB SEND command (Server) 692
WEB STARTBROWSE FORMFIELD command 708
CLNTIPFAMILY option
EXTRACT TCPIP command 183
CLOSESTATUS option
WEB CONVERSE command 640
WEB SEND command (Client) 701
WEB SEND command (Server) 692
CLRPARTN option
HANDLE AID command 264
CMDSEC option
ASSIGN command 47
CNAMELENGTH option
EXTRACT TCPIP command 183
CNOTCOMPL option
SEND (non-z/OS Communications Server) command 496
SEND (z/OS Communications Server) command 491
COBOL
argument values 3
translated code 12
CODEPAGE option
WEB OPEN command 658
CODEPAGEERR condition
GET CONTAINER (CHANNEL) command 238
GET64 CONTAINER command 261
PUT CONTAINER (CHANNEL) command 360
PUT64 CONTAINER command 364
WSACONTEXT BUILD command 738
COLOR operand
DFHMDF 857
DFHMDI 868
DFHMSD 877
COLOR option
ASSIGN command 47
COLUMN operand
DFHMDI 868
column value
DFHMDI 868
command language translator
translated code 13
commands
format, arguments 1
scheduling 21
commands *(continued)*
security 21
spool 22
TCP/IP 22
temporary storage control 22
COMMAREA option
ADDRESS command 31
INVOKE APPLICATION command 279
LINK command 326
RETURN command 460
XCTL command 748
common buffer, output to, 2980 425
common programming interface communications (CPI communications) 847
COMMONNAME option
EXTRACT CERTIFICATE command 178
COMMONNAMLEN option
EXTRACT CERTIFICATE command 178
COMPAREMAX option
GET COUNTER command 239
GET DCOUNTER command 239
UPDATE COUNTER command 610
UPDATE DCOUNTER command 610
COMPAREMIN option
GET COUNTER command 239
GET DCOUNTER command 239
UPDATE COUNTER command 610
UPDATE DCOUNTER command 610
COMPLETE mode, of an activity 270
COMPLETE option
DUMP TRANSACTION command 154
COMPOSITE option
GETNEXT EVENT command 255
INQUIRE EVENT command 274
COMPSTATUS option
CHECK ACQPROCESS command 75
CHECK ACTIVITY command 77
FETCH CHILD command 191, 192
INQUIRE ACTIVITYID command 270
CONFIRM option
GDS SEND command 229
SEND (z/OS Communications Server) command 491
CONNECT PROCESS command 80
CONSISTENT option
READ command 375
READNEXT command 388
READPREV command 399
CONSOLE option
ISSUE ABORT command 289
ISSUE END command 297
ISSUE SEND command 318
ISSUE WAIT command 322
console support commands 20
constants
AID values, DFHAID 854
attribute values, DFHBMSCA 849
for 3270 attributes 849
for examining EIBAID field 854
for MSR control values 852
for printer format controls 849
constants *(continued)*
MSR control, DFHtex read 852
printer control values, DFHBMSCA 849
container commands
DELETE CHANNEL 129
DELETE CONTAINER (BTS) 130
DELETE CONTAINER (CHANNEL) 132
ENDBROWSE CONTAINER 161
ENDBROWSE CONTAINER (CHANNEL) 161
GET CONTAINER (BTS) 232
GET CONTAINER (CHANNEL) 234
GET64 CONTAINER 258
GETNEXT CONTAINER 253
GETNEXT CONTAINER (CHANNEL) 254
INQUIRE CONTAINER 272
MOVE CONTAINER (BTS) 343
MOVE CONTAINER (CHANNEL) 345
PUT CONTAINER (BTS) 354
PUT CONTAINER (CHANNEL) 357
PUT64 CONTAINER 361
QUERY CHANNEL 365
STARTBROWSE CONTAINER 588
STARTBROWSE CONTAINER (CHANNEL) 590
CONTAINER option
DELETE CONTAINER (BTS) command 131
DELETE CONTAINER (CHANNEL) command 132
GET CONTAINER (BTS) command 232
GET CONTAINER (CHANNEL) command 236
GET64 CONTAINER command 259
GETNEXT CONTAINER command 254
GETNEXT CONTAINER command (CHANNEL) 255
INQUIRE CONTAINER command 272
MOVE CONTAINER (BTS) command 344
MOVE CONTAINER (CHANNEL) command 347
PUT CONTAINER (BTS) command 355
PUT CONTAINER (CHANNEL) command 358
PUT64 CONTAINER command 362
WEB CONVERSE command 641
WEB SEND command (Client) 702
WEB SEND command (Server) 693
CONTAINERCNT option
QUERY CHANNEL command 365
CONTAINERERR condition
DELETE CONTAINER (BTS) command 131
DELETE CONTAINER (CHANNEL) command 132
GET CONTAINER (BTS) command 233

CONTAINERERR condition *(continued)*
GET CONTAINER (CHANNEL) command 239
GET64 CONTAINER command 262
INQUIRE CONTAINER command 273
MOVE CONTAINER (BTS) command 345
MOVE CONTAINER (CHANNEL) command 347
PUT CONTAINER (BTS) command 356
PUT CONTAINER (CHANNEL) command 360
PUT64 CONTAINER command 364
WEB CONVERSE command 648
WEB RECEIVE command (Client) 687
WEB RECEIVE command (Server) 681
WEB SEND command (Client) 706
WEB SEND command (Server) 696
context-switching
described 333, 336, 475
CONTROL option
QUERY SECURITY command 369
CONVDATA option
GDS CONNECT PROCESS command 213
GDS EXTRACT ATTRIBUTES command 216
GDS FREE command 218
GDS ISSUE ABEND command 220
GDS ISSUE CONFIRMATION command 221
GDS ISSUE ERROR command 222
GDS ISSUE PREPARE command 224
GDS ISSUE SIGNAL command 225
GDS RECEIVE command 227
GDS SEND command 229
GDS WAIT command 231
CONVERSE (2260) command 97
CONVERSE (3270 logical) command 86
CONVERSE (3600-3601) command 86
CONVERSE (3600-3614) command 87
CONVERSE (3650 interpreter) command 88
CONVERSE (3650-3270) command 88
CONVERSE (3650-3653) command 89
CONVERSE (3650-3680) command 89
CONVERSE (3767) command 90
CONVERSE (3770) command 90
CONVERSE (3790 3270-display) command 91
CONVERSE (3790 full-function or inquiry) command 91
CONVERSE (APPC) command 83
CONVERSE (default) command 83
CONVERSE (LUTYPE2/LUTYPE3) command 83
CONVERSE (LUTYPE4) command 84
CONVERSE (LUTYPE6.1) command 84
CONVERSE (MRO) command 97
CONVERSE (non-z/OS Communications Server default) command 96
CONVERSE (SCS) command 85
CONVERSE option
ISSUE LOAD command 306
CONVERSE WEB command 637
converse with terminal or LU 841
CONVERTST option
GET CONTAINER (CHANNEL) command 236
GET64 CONTAINER command 259
CONVERTTIME command 102
CONVID option
CONNECT PROCESS command 80
CONVERSE (non-z/OS Communications Server) command 98
CONVERSE (z/OS Communications Server) command 92
EXTRACT ATTACH (LUTYPE6.1) command 168
EXTRACT ATTACH (MRO) command 172
EXTRACT ATTRIBUTES (APPC) command 174
EXTRACT ATTRIBUTES (MRO) command 176
EXTRACT PROCESS command 180
FREE (APPC) command 201
FREE (LUTYPE6.1) command 202
FREE (MRO) command 203
GDS ALLOCATE command 210
GDS CONNECT PROCESS command 214
GDS EXTRACT ATTRIBUTES command 215
GDS EXTRACT PROCESS command 217
GDS FREE command 219
GDS ISSUE ABEND command 220
GDS ISSUE CONFIRMATION command 221
GDS ISSUE ERROR command 223
GDS ISSUE PREPARE command 224
GDS ISSUE SIGNAL command 225
GDS RECEIVE command 227
GDS SEND command 229
GDS WAIT command 231
ISSUE ABEND command 287
ISSUE CONFIRMATION command 292
ISSUE ERROR command 304
ISSUE PREPARE command 310
ISSUE SIGNAL (APPC) command 320
ISSUE SIGNAL (LUTYPE6.1) command 321
POINT command 348
RECEIVE (z/OS Communications Server) command 419
SEND (z/OS Communications Server) command 491
WAIT CONVID command 623
WAIT TERMINAL command 632
copy displayed information 845
copybooks
DFHAID 854
DFHBMSCA 849
DFHEIBLK 13
DFHMSRCA 852
COUNTER option
DEFINE COUNTER command 108
DEFINE DCOUNTER command 108
DELETE COUNTER command 133
GET COUNTER command 239
QUERY COUNTER command 365
REWIND COUNTER command 463
UPDATE COUNTER command 610
COUNTRY option
EXTRACT CERTIFICATE command 178
COUNTRYLEN option
EXTRACT CERTIFICATE command 178
CPI communications (SAA) 847
create a journal record 323, 722
CTLCHAR option
CONVERSE (non-z/OS Communications Server) command 98
CONVERSE (z/OS Communications Server) command 92
ISSUE COPY (3270 logical) command 293
SEND (non-z/OS Communications Server) command 496
SEND (z/OS Communications Server) command 491
CTRL operand
DFHMDI 868
DFHMSD 877
CURRENT option
SEND PAGE command 515
CURSLOC operand
DFHMDI 868
DFHMSD 877
cursor address 846
CURSOR option
RECEIVE MAP MAPPINGDEV command 434
SEND CONTROL command 500
SEND MAP command 505
SEND MAP MAPPINGDEV command 512
SEND TEXT command 519
cursor position
terminal control 846
CVDA (CICS-value data area)
command format 3
listed in numerical and alphabetical order 787
CVDA options
ACTION
WRITE OPERATOR command 727
ALTER
QUERY SECURITY command 369
ASRAKEY
ASSIGN command 44
ASRASPC
ASSIGN command 45
CONTROL
QUERY SECURITY command 369
LOGMESSAGE
QUERY SECURITY command 369
MAXLIFETIME
DEQ 141

CVDA options *(continued)*
MAXLIFETIME *(continued)*
ENQ 165
PURGEABILITY
WAIT EXTERNAL 627
WAITCICS 634
READ
QUERY SECURITY command 369
STATE 94, 100, 420, 429, 492, 498
ALLOCATE (APPC) 35
ALLOCATE (MRO) 39
CONNECT PROCESS 81
EXTRACT ATTRIBUTES (APPC) 174
EXTRACT ATTRIBUTES (MRO) 176
FREE (APPC) 201
FREE (MRO) 204
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 221
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 225
GDS RECEIVE 227
GDS SEND 229
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 304
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
WAIT CONVID 623
UPDATE
QUERY SECURITY command 371
CVDA values 199
ALLOCATED
ALLOCATE (APPC) 35
ALLOCATE (MRO) 39
CONNECT PROCESS 81
EXTRACT ATTRIBUTES (APPC) 175
EXTRACT ATTRIBUTES (MRO) 176
FREE (APPC) 201
FREE (MRO) 204
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 221
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 225
GDS RECEIVE 227
GDS SEND 229
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 304
ISSUE PREPARE 310
CVDA values *(continued)*
ALLOCATED *(continued)*
ISSUE SIGNAL (APPC) 320
RECEIVE (MRO) command 429
RECEIVE (z/OS Communications Server) command 420
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 492
WAIT CONVID 623
ALTERABLE
QUERY SECURITY command 369
ASSERTED
EXTRACT TCPIP command 182
ATTLSAWARE
EXTRACT TCPIP command 182, 185
AUTOAUTH
EXTRACT TCPIP command 182
AUTOREGISTER
EXTRACT TCPIP command 182
BASE64
BIF DIGEST command 60
BASESPACE
ASSIGN command 46
BASICAUTH
EXTRACT TCPIP command 182
WEB CONVERSE command (Client) 640
WEB SEND command (Client) 700
BINARY
BIF DIGEST command 60
CERTIFICAUTH
EXTRACT TCPIP command 182
CHUNKEND
WEB SEND command (Client) 701
WEB SEND command (Server) 692
CHUNKNO
WEB SEND command (Client) 701
WEB SEND command (Server) 692
CHUNKYES
WEB SEND command (Client) 701
WEB SEND command (Server) 692
CICSEXECKEY
ASSIGN command 44
CLICONVERT
WEB CONVERSE command 647
WEB RECEIVE command (Client) 684
WEB SEND command (Client) 702
CLIENTAUTH
EXTRACT TCPIP command 185
CLOSE
WEB CONVERSE command 640
WEB SEND command (Client) 701
WEB SEND command (Server) 692
CVDA values *(continued)*
CONFFREE
CONNECT PROCESS 81
EXTRACT ATTRIBUTES (APPC) 175
FREE (APPC) 201
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 221
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 225
GDS RECEIVE 228
GDS SEND 229
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 304
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (z/OS Communications Server) command 421
SEND (z/OS Communications Server) command 492
WAIT CONVID 623
CONFRECEIVE
CONNECT PROCESS 81
EXTRACT ATTRIBUTES (APPC) 175
FREE (APPC) 201
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 221
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 225
GDS RECEIVE 228
GDS SEND 229
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 304
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (z/OS Communications Server) command 421
SEND (z/OS Communications Server) command 492
WAIT CONVID 623
CONFSEND
CONNECT PROCESS 81
EXTRACT ATTRIBUTES (APPC) 175
FREE (APPC) 201
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220

CVDA values *(continued)*
CONFSEND *(continued)*
GDS ISSUE CONFIRMATION 221
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 225
GDS RECEIVE 228
GDS SEND 230
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 304
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (z/OS Communications Server) command 421
SEND (z/OS Communications Server) command 492
WAIT CONVID 623
CRITICAL
WRITE OPERATOR command 727
CTRLABLE
QUERY SECURITY command 369
DELETE
WEB CONVERSE command 642, 704
DOCDELETE
WEB CONVERSE command 641
WEB SEND command (Client) 703
WEB SEND command (Server) 693
EVENTUAL
WEB SEND command (Server) 691
WRITE OPERATOR command 727
EXPECT
WEB CONVERSE command 639
WEB SEND command (Client) 700
FREE
CONNECT PROCESS 81
EXTRACT ATTRIBUTES (APPC) 175
EXTRACT ATTRIBUTES (MRO) 176
FREE (APPC) 201
FREE (MRO) 204
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 222
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 225
GDS RECEIVE 228
GDS SEND 230
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 304
CVDA values *(continued)*
FREE *(continued)*
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (MRO) command 429
RECEIVE (z/OS Communications Server) command 421
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 492
WAIT CONVID 623
GET
WEB CONVERSE command 642
WEB SEND command (Client) 704
HEAD
WEB CONVERSE command 642
WEB SEND command (Client) 704
HEX
BIF DIGEST command 60
HOSTNAME
WEB EXTRACT or EXTRACT WEB command 188, 654
WEB PARSE URL command 663
HTTP
WEB EXTRACT or EXTRACT WEB command 189, 655
WEB OPEN command 660
HTTPNO
WEB EXTRACT or EXTRACT WEB command 189, 655
WEB RECEIVE command (Server) 680
HTTPS
WEB EXTRACT or EXTRACT WEB command 189, 655
WEB OPEN command 660
HTTPYES
WEB EXTRACT or EXTRACT WEB command 189, 655
WEB RECEIVE command (Server) 680
IMMEDIATE
WEB SEND command (Server) 691
WRITE OPERATOR command 727
IPV4
EXTRACT TCPIP command 183, 185
WEB EXTRACT or EXTRACT WEB command 188, 654
WEB PARSE URL command 663
IPV6
EXTRACT TCPIP command 183, 185
WEB EXTRACT or EXTRACT WEB command 188, 654
WEB PARSE URL command 663
LOG
QUERY SECURITY command 369
NOAUTHENTIC
EXTRACT TCPIP command 182
NOCLICONVERT
WEB CONVERSE command 648
CVDA values *(continued)*
NOCLICONVERT *(continued)*
WEB RECEIVE command (Client) 684
WEB SEND command (Client) 702
NOCLOSE
WEB CONVERSE command 640
WEB SEND command (Client) 702
WEB SEND command (Server) 693
NOCONVERT
GET CONTAINER (CHANNEL) command 236
GET64 CONTAINER command 259
NODOCDELETE
WEB CONVERSE command 641
WEB SEND command (Client) 703
WEB SEND command (Server) 693
NOINCONVERT
WEB CONVERSE command 647
NOLOG
QUERY SECURITY command 369
NONCICS
ASSIGN command 45
NONE
WEB CONVERSE command (Client) 640
WEB SEND command (Client) 700
NOOUTCONVERT
WEB CONVERSE command 647
NOSRVCONVERT
WEB RECEIVE command (Server) 679
WEB SEND command (Server) 695
NOSSL
EXTRACT TCPIP command 185
NOTALTERABLE
QUERY SECURITY command 369
NOTAPPLIC
ASSIGN command 45, 46
EXTRACT TCPIP command 183, 185
WEB EXTRACT or EXTRACT WEB command 188, 654
NOTCTRLABLE
QUERY SECURITY command 369
NOTPURGEABLE
WAIT EXTERNAL 627
WAITCICS 634
NOTREADABLE
QUERY SECURITY command 369
NOTSUPPORTED
EXTRACT TCPIP command 182
NOTUPDATABLE
QUERY SECURITY command 371
OPTIONS
WEB CONVERSE command 642
WEB SEND command (Client) 704

CVDA values *(continued)*
PENDFREE
CONNECT PROCESS 81
EXTRACT ATTRIBUTES (APPC) 175
EXTRACT ATTRIBUTES (MRO) 176
FREE (APPC) 201
FREE (MRO) 204
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 222
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 226
GDS RECEIVE 228
GDS SEND 230
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 304
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (MRO) command 429
RECEIVE (z/OS Communications Server) command 421
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 492
WAIT CONVID 623
PENDRECEIVE
CONNECT PROCESS 81
EXTRACT ATTRIBUTES (APPC) 175
FREE (APPC) 201
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 222
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 226
GDS RECEIVE 228
GDS SEND 230
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 304
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (z/OS Communications Server) command 421
SEND (z/OS Communications Server) command 492
WAIT CONVID 623
PURGEABLE
WAIT EXTERNAL 627
WAITCICS 634
CVDA values *(continued)*
PUT
WEB CONVERSE command 642
WEB SEND command (Client) 704
READABLE
QUERY SECURITY command 369
RECEIVE
CONNECT PROCESS 81
CONVERSE command (non-z/OS Communications Server) 100
EXTRACT ATTRIBUTES (APPC) 175
EXTRACT ATTRIBUTES (MRO) 176
FREE (APPC) 201
FREE (MRO) 204
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 222
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 226
GDS RECEIVE 228
GDS SEND 230
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 304
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (MRO) command 429
RECEIVE (z/OS Communications Server) command 421
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 492
WAIT CONVID 623
REQUIRED
EXTRACT TCPIP command 182
RFC1123
FORMATTIME command 198
RFC3339
FORMATTIME command 198
ROLLBACK
CONNECT PROCESS 81
CONVERSE command (non-z/OS Communications Server) 100
EXTRACT ATTRIBUTES (APPC) 175
EXTRACT ATTRIBUTES (MRO) 176
FREE (APPC) 201
FREE (MRO) 204
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 222
GDS ISSUE ERROR 223
CVDA values *(continued)*
ROLLBACK *(continued)*
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 226
GDS RECEIVE 228
GDS SEND 230
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 305
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (MRO) command 429
RECEIVE (z/OS Communications Server) command 421
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 492
WAIT CONVID 623
SEND
CONNECT PROCESS 81
CONVERSE command (non-z/OS Communications Server) 100
EXTRACT ATTRIBUTES (APPC) 175
EXTRACT ATTRIBUTES (MRO) 176
FREE (APPC) 202
FREE (MRO) 204
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 222
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 226
GDS RECEIVE 228
GDS SEND 230
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 305
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (MRO) command 429
RECEIVE (z/OS Communications Server) command 421
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 492
WAIT CONVID 623
SRVCONVERT
WEB RECEIVE command (Server) 679
WEB SEND command (Server) 695
SSL
EXTRACT TCPIP command 185
SUBSPACE
ASSIGN command 46
SUPPORTED
EXTRACT TCPIP command 184

CVDA values *(continued)*
SYNCFREE
CONNECT PROCESS 81
CONVERSE command (non-z/OS Communications Server) 100
EXTRACT ATTRIBUTES (APPC) 175
EXTRACT ATTRIBUTES (MRO) 176
FREE (APPC) 202
FREE (MRO) 204
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 222
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 226
GDS RECEIVE 228
GDS SEND 230
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 305
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (MRO) command 429
RECEIVE (z/OS Communications Server) command 421
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 492
WAIT CONVID 624
SYNCRECEIVE
CONNECT PROCESS 81
CONVERSE command (non-z/OS Communications Server) 100
EXTRACT ATTRIBUTES (APPC) 175
EXTRACT ATTRIBUTES (MRO) 176
FREE (APPC) 202
FREE (MRO) 204
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 222
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 226
GDS RECEIVE 228
GDS SEND 230
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 305
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (MRO) command 429
CVDA values *(continued)*
SYNCRECEIVE *(continued)*
RECEIVE (z/OS Communications Server) command 421
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 493
WAIT CONVID 624
SYNCSEND
CONNECT PROCESS 81
CONVERSE command (non-z/OS Communications Server) 100
EXTRACT ATTRIBUTES (APPC) 175
EXTRACT ATTRIBUTES (MRO) 176
FREE (APPC) 202
FREE (MRO) 204
GDS ALLOCATE 211
GDS CONNECT PROCESS 214
GDS EXTRACT ATTRIBUTES 216
GDS FREE 219
GDS ISSUE ABEND 220
GDS ISSUE CONFIRMATION 222
GDS ISSUE ERROR 223
GDS ISSUE PREPARE 224
GDS ISSUE SIGNAL 226
GDS RECEIVE 228
GDS SEND 230
GDS WAIT 231
ISSUE ABEND command 287
ISSUE CONFIRMATION 292
ISSUE ERROR 305
ISSUE PREPARE 310
ISSUE SIGNAL (APPC) 320
RECEIVE (MRO) command 429
RECEIVE (z/OS Communications Server) command 421
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 493
WAIT CONVID 624
TASK
DEQ 141
ENQ 165
TRACE
WEB CONVERSE command 642
WEB SEND command (Client) 704
UOW
DEQ 141
ENQ 165
UPDATABLE
QUERY SECURITY command 371
USEREXECKEY
ASSIGN command 45
CWA option
ADDRESS command 32
CWALENG option
ASSIGN command 47

D

data
passing to new tasks 561
DATA operand
DFHMDI 868
DFHMSD 877
DATA option
FREEMAIN command 206
FREEMAIN64 command 209
data sets
add records to 290
interrogating 312
processing termination 297
read records from 313
update records 315
data tables
CICS/user-maintained/coupling facility
DELETE command 120
ENDBR command 158
READ command 372
READNEXT command 384
READPREV command 395
RESETBR command 447
REWRITE command 465
STARTBR command 580
UNLOCK command 606
WRITE command 715
data to output device, sending 317
data-area argument
CICS command format 3
definition 1
data-area64
command format 3
data-value argument
CICS command format 3
definition 1
data, deleting
file control records 120
named counter 133
temporary storage queues 138
transient data queues 136
DATA1 option
MONITOR command 341
DATA2 option
MONITOR command 341
DATALENGTH option
INQUIRE CONTAINER command 273
LINK command 326
DATAONLY option
DOCUMENT RETRIEVE command 150
SEND MAP command 505
SEND MAP MAPPINGDEV command 512
DATAPOINTER option
FREEMAIN command 206
FREEMAIN64 command 209
DATASTR option
BUILD ATTACH (LUTYPE6.1) command 61
BUILD ATTACH (MRO) command 64
EXTRACT ATTACH (LUTYPE6.1) command 169

DATASTR option *(continued)*
EXTRACT ATTACH (MRO) command 172
DATATYPE option
PUT CONTAINER (CHANNEL) command 358
PUT64 CONTAINER command 362
DATCONTAINERoption
TRANSFORM XMLTODATA command 604
DATE option
FORMATTIME command 197
DATEFORM option
FORMATTIME command 197
DATESEP option
FORMATTIME command 197
DATESTRING option
CONVERTTIME command 103
FORMATTIME command 197
DAYCOUNT option
FORMATTIME command 197
DAYOFMONTH option
DEFINE TIMER command 116
FORMATTIME command 197
DAYOFWEEK option
FORMATTIME command 198
DAYOFYEAR option
DEFINE TIMER command 116
DAYS option
DEFINE TIMER command 116
DAYSLEFT option
VERIFY PASSWORD command 615
VERIFY PHRASE command 617
DCOUNTER option
DELETE DCOUNTER command 133
GET DCOUNTER command 239
QUERY DCOUNTER command 365
REWIND DCOUNTER command 463
UPDATE DCOUNTER command 610
DDMMYY option
FORMATTIME command 198
DDMMYYYY option
FORMATTIME command 198
DEBKEY option
READ command 375
STARTBR command 582
DEBREC option
READ command 375
STARTBR command 582
DEFAULT option
CONVERSE (non-z/OS Communications Server) command 98
CONVERSE (z/OS Communications Server) command 92
SEND (non-z/OS Communications Server) command 496
SEND (z/OS Communications Server) command 491
SEND CONTROL command 500
SEND MAP command 505
SEND TEXT command 520
SEND TEXT NOEDIT command 528
DEFINE ACTIVITY command 104
DEFINE COMPOSITE EVENT command 106
DEFINE COUNTER command 108
DEFINE DCOUNTER command 108
DEFINE INPUT EVENT command 111
DEFINE PROCESS command 112
DEFINE TIMER command 115
DEFRESP option
CONVERSE (non-z/OS Communications Server) command 98
CONVERSE (z/OS Communications Server) command 92
ISSUE ADD command 290
ISSUE ERASE command 301
ISSUE REPLACE command 315
ISSUE SEND command 318
SEND (non-z/OS Communications Server) command 497
SEND (z/OS Communications Server) command 491
DEFSCRNHT option
ASSIGN command 47
DEFSCRNWD option
ASSIGN command 47
DELAY command 117
delay processing, task 117
DELETE ACTIVITY command 128
DELETE CHANNEL command 129
DELETE command 120
DELETE CONTAINER (BTS) command 130
DELETE CONTAINER (CHANNEL) command 132
DELETE COUNTER command 133
DELETE DCOUNTER command 133
DELETE EVENT command 135
delete loaded program 438
DELETE option
SPOOLCLOSE command 548
delete records
batch data interchange records 301
DELETE TIMER command 136
DELETEQ TD command 136
DELETEQ TS command 138
deleting data
named counter 133
temporary storage queues 138
transient data queues 136
DELIMITER option
ASSIGN command 47
DEQ command 139
dequeue from resource 139
DESTCOUNT option
ASSIGN command 47
DESTID option
ASSIGN command 47
ISSUE ABORT command 289
ISSUE ADD command 290
ISSUE END command 297
ISSUE ERASE command 301
ISSUE NOTE command 307
ISSUE QUERY command 312
ISSUE REPLACE command 315
ISSUE SEND command 318
ISSUE WAIT command 322
DESTIDLENG option
ASSIGN command 48
ISSUE ABORT command 289
DESTIDLENG option *(continued)*
ISSUE ADD command 290
ISSUE END command 297
ISSUE ERASE command 301
ISSUE NOTE command 307
ISSUE QUERY command 312
ISSUE REPLACE command 315
ISSUE SEND command 318
ISSUE WAIT command 322
destruction of activities 128
DET value
DFHMDF 857
DFH2980 structure 425
DFHAID attention identifier list 854
DFHBMSCA, standard attribute and printer control character list, BMS 849
DFHEAI interface processor 13
DFHECALL macro 13
DFHEIBLK copybook 13
DFHEIEND macro 13
DFHEIENT macro description 13
DFHEIGBL macro 13
DFHEIRET macro 13
DFHEISTG macro 13
DFHMDF macro 857
DFHMDI macro 868
DFHMIRS 328
DFHMSD macro 877
DFHMSRCA, MSR control value constants 852
DFHPDI macro 888
DFHPSD macro 889
DFHRESP, built-in function 11
diagnostic services commands 20
DIGESTTYPE option
BIF DIGEST command 60
DISABLED condition
DELETE command 120
DELETEQ TD command 137
READ command 379
READQ TD command 407
STARTBR command 584
UNLOCK command 608
WRITE command 718
WRITEQ TD command 729
disconnect a switched line 841
display-device operations
attention identifier (AID) 846
attention identifier list, DFHAID 854
copy displayed information 845
cursor address 846
erase all unprotected fields 845
input operation without data 846
pass control on receipt of an AID 264
print displayed information 845
standard attribute and printer control character list, DFHBMSCA 849
terminal 844
DOCSIZE option
DOCUMENT INSERT command 147
DOCSTATUS option
WEB CONVERSE command 641
WEB SEND command (Client) 703
WEB SEND command (Server) 693

DOCTOKEN option
DOCUMENT RETRIEVE command 150
DOCUMENT SET command 152
WEB CONVERSE command 641
WEB RETRIEVE command 689
WEB SEND command (Client) 703
WEB SEND command (Server) 693
document
adding symbols to symbol table 151
creating 141
deleting 145
DOCUMENT CREATE command 141
DOCUMENT DELETE command 145
DOCUMENT INSERT command 146
DOCUMENT option
DOCUMENT INSERT command 147
DOCUMENT RETRIEVE command 149
document services
commands 20
DOCUMENT SET command 151
DORMANT mode, of an activity 270
DRK value
DFHMDF 857
DS3270 option
ASSIGN command 48
DSATTS operand
DFHMDI 868
DFHMSD 877
DSECT operand
DFHMSD 877
DSECT value
DFHMSD 877
DSSCS option
ASSIGN command 48
DSSTAT condition
ISSUE RECEIVE command 314
DUMP TRANSACTION command 153
DUMPCODE option
DUMP TRANSACTION command 154
DUMPID option
DUMP TRANSACTION command 154
DUPKEY condition
DELETE command 120
READ command 380
READNEXT command 392
READPREV command 402
DUPREC condition
REWRITE command 467
WRITE command 718
dynamic allocation 552

E

ECADDR option
WAIT EVENT command 625
ECBLIST option
WAIT EXTERNAL command 627
WAITCICS command 633
EDF, execution diagnostic facility 561, 576
EIB fields
EIBAID 755
EIBATT 755
EIBCALEN 755
EIB fields *(continued)*
EIBCOMPL 755
EIBCONF 755
EIBCPOSN 755
EIBDATE 755
EIBDS 755
EIBEOC 755
EIBERR 755
EIBERRCD 755
EIBFMH 755
EIBFN 755
EIBFREE 755
EIBNODAT 755
EIBRCODE 755
EIBRECV 755
EIBREQID 755
EIBRESP 755
EIBRESP2 755
EIBRLDBK 755
EIBRSRCE 755
EIBSIG 755
EIBSYNC 755
EIBSYNRB 755
EIBTASKN 755
EIBTIME 755
EIBTRMID 755
EIBTRNID 755
ILLOGIC condition
EIBRCODE 755
IOERR condition
EIBRCODE 755
EIB option
ADDRESS command 32
EIBAID 846
examining contents of field 854
EIBRCODE 755
EIBRESP 755
ELEMNAME option
TRANSFORM XMLTODATA command 604
ELEMNAMELEN option
TRANSFORM XMLTODATA command 604
ELEMNS option
TRANSFORM XMLTODATA command 604
ELEMNSLEN option
TRANSFORM XMLTODATA command 604
END condition
GETNEXT ACTIVITY command 253
GETNEXT CONTAINER (CHANNEL) command 255
GETNEXT CONTAINER command 254
GETNEXT EVENT command 256
GETNEXT PROCESS command 257
RETRIEVE REATTACH EVENT command 457
RETRIEVE SUBEVENT command 459
ENDACTIVITY option
RETURN command 461
ENDBR command 158
ENDBROWSE ACTIVITY command 160
ENDBROWSE CONTAINER (CHANNEL) command 161
ENDBROWSE CONTAINER command 161
ENDBROWSE EVENT command 162
ENDBROWSE PROCESS command 163
ENDDATA condition
RETRIEVE command 455
ENDFILE condition
READNEXT command 392
READPREV command 402
SPOOLREAD command 557
WEB READNEXT FORMFIELD command 671
WEB READNEXT HTTPHEADER command 672
WEB READNEXT QUERYPARM command 674
ENDFILE option
ISSUE ENDOUTPUT command 299
ENDINPT condition
RECEIVE (non-z/OS Communications Server) command 429
ENDOUTPUT option
ISSUE ENDFILE command 299
English and katakana characters, mixed 98, 431, 436
ENQ command 163
ENQBUSY condition
ENQ command 165
ensuring terminal operation has completed 632
ENTER option
HANDLE AID command 264
ENTER TRACEID command
monitoring aspects replaced by MONITOR command 341
tracing aspects replaced by ENTER TRACENUM command 167
ENTER TRACENUM command 166
ENTRY option
LOAD command 339
entry to assembler-language program 13
ENTRYNAME option
MONITOR command 342
ENVDEFERR condition
RETRIEVE command 455
environment services
commands 20
EOC condition
ALLOCATE (LUTYPE6.1) command 38
CONVERSE (non-z/OS Communications Server) command 101
CONVERSE (z/OS Communications Server) command 95
ISSUE RECEIVE command 314
RECEIVE (non-z/OS Communications Server) command 429
RECEIVE (z/OS Communications Server) command 421
RECEIVE MAP command 432
RECEIVE PARTN command 437
WAIT TERMINAL command 632
EODS condition
CONVERSE (z/OS Communications Server) command 95
ISSUE RECEIVE command 314

EODS condition *(continued)*
RECEIVE (z/OS Communications Server) command 421
RECEIVE MAP command 432
RECEIVE PARTN command 437
EOF condition
CONVERSE (non-z/OS Communications Server) command 101
RECEIVE (non-z/OS Communications Server) command 429
EQUAL option
READ command 375
RESETBR command 448
STARTBR command 582
equated symbols 7
erase all unprotected fields 845
ERASE option
CONVERSE (non-z/OS Communications Server) command 98
CONVERSE (z/OS Communications Server) command 92
SEND (non-z/OS Communications Server) command 497
SEND (z/OS Communications Server) command 491
SEND CONTROL command 500
SEND MAP command 505
SEND MAP MAPPINGDEV command 512
SEND TEXT command 520
SEND TEXT NOEDIT command 528
ERASEAUP option
SEND CONTROL command 501
SEND MAP command 506
SEND MAP MAPPINGDEV command 512
ERRORMSG option
ASSIGN command 48
ERRORMSGELN option
ASSIGN command 48
ERRTERM option
ROUTE command 471
ESDS (entry-sequenced data set)
DELETE command 120
READ 377, 379
READNEXT 392
READPREV 402
RESETBR 449
STARTBR 584
STARTBR command 583
WRITE command 716
ESM
ACEE pointer 31
QUERY SECURITY command, NOTFND condition 372
QUERY SECURITY command, RESCLASS option 370
USERNAME 56
ESM, external security manager 561, 576
ESMAPPNAME option
REQUEST PASSTICKET 441, 443
ESMREASON option
CHANGE PASSWORD command 73
CHANGE PHRASE command 71
SIGNON command 534
ESMREASON option *(continued)*
SIGNON TOKEN command 538
VERIFY PASSWORD command 615
VERIFY PHRASE command 618
VERIFY TOKEN command 621
ESMRESP option
CHANGE PASSWORD command 73
CHANGE PHRASE command 71
REQUEST PASSTICKET 442, 444
SIGNON command 534
SIGNON TOKEN command 538
VERIFY PASSWORD command 615
VERIFY PHRASE command 618
VERIFY TOKEN command 621
EVENT option
ADD SUBEVENT command 30
DEFINE ACTIVITY command 104
DEFINE COMPOSITE EVENT command 107
DEFINE INPUT EVENT command 111
DEFINE TIMER command 116
DELETE EVENT command 135
GETNEXT EVENT command 255
INQUIRE ACTIVITYID command 270
INQUIRE EVENT command 274
INQUIRE TIMER command 277
REMOVE SUBEVENT command 440
RETRIEVE REATTACH EVENT command 457
RETRIEVE SUBEVENT command 458
SIGNAL EVENT command 531
TEST EVENT command 597
event processing commands 20
event-related commands
CHECK TIMER 79
DEFINE COMPOSITE EVENT 106
DEFINE INPUT EVENT 111
DEFINE TIMER 115
DELETE EVENT 135
DELETE TIMER 136
ENDBROWSE EVENT 162
FORCE TIMER 194
GETNEXT EVENT 255
INQUIRE EVENT 274
INQUIRE TIMER 276
REMOVE SUBEVENT 439
RETRIEVE REATTACH EVENT 456
RETRIEVE SUBEVENT 458
STARTBROWSE EVENT 591
TEST EVENT 597
EVENTERR condition
ADD SUBEVENT command 30
DEFINE ACTIVITY command 105
DEFINE COMPOSITE EVENT command 107
DEFINE INPUT EVENT command 112
DEFINE TIMER command 117
DELETE EVENT command 135
INQUIRE EVENT command 275
LINK ACQPROCESS command 333
LINK ACTIVITY command 337
REMOVE SUBEVENT command 440
EVENTERR condition *(continued)*
RETRIEVE SUBEVENT command 459
RUN command 478
SIGNAL EVENT command 531
TEST EVENT command 597
events, timer
control area, timer 350
monitoring point 341
waiting for 624
EVENTTYPE option
GETNEXT EVENT command 256
INQUIRE EVENT command 274
RETRIEVE REATTACH EVENT command 457
RETRIEVE SUBEVENT command 458
EWASUPP option
ASSIGN command 48
EXACTMATCH option
INVOKE APPLICATION command 279
examples
using the ADDRESS SET command 33
using the ASKTIME command 40
using the ASSIGN command 56
using the BIF DEEDIT command 59
using the CANCEL command 25
using the DELAY command 117
using the DELETE command 120
using the DEQ command 141
using the DUMP TRANSACTION command 153
using the ENQ command 166
using the ENTER TRACENUM command 168
using the FORMATTIME command 200
using the FREEMAIN command 206
using the GETMAIN command 248, 251
using the HANDLE ABEND command 264
using the HANDLE AID command 266
using the HANDLE CONDITION command 267
using the LINK command 332
using the LOAD command 340
using the MONITOR command 342
using the POST command 353
using the READ command 384
using the READQ TD command 408
using the READQ TS command 412
using the RELEASE command 439
using the RETRIEVE command 456
using the REWRITE command 470
using the START ATTACH command 572
using the START BREXIT command 573
using the START command 561
using the WAIT EVENT command 625
using the WAIT EXTERNAL command 628

examples *(continued)*
using the WAIT JOURNALNAME command 630
using the WAITCICS command 633
using the WRITE command 722
using the WRITE JOURNALNAME command 725
using the XCTL command 750
EXCEPTION option
ENTER TRACENUM command 167
exception support commands 20
exclusive control release, UNLOCK command 606
EXEC CICS command format 1
EXEC CICS commands
Function codes 774
Response Codes 774
execution diagnostic facility (EDF) 561, 576
exit from ASM program 13
exit, abnormal termination recovery 262
expiration time, notification when reached 349
EXPIRED condition
DELAY command 117
POST command 352
WRITE OPERATOR command 728
EXPIRYTIME option
VERIFY PASSWORD command 615
VERIFY PHRASE command 618
EXTATT operand
DFHMDI 868
DFHMSD 877
EXTDS option
ASSIGN command 48
extended relative byte address (XRBA) 120
external security manager (ESM) 369, 561, 576
EXTRACT ATTACH (LUTYPE6.1) command 168
EXTRACT ATTACH (MRO) command 171
EXTRACT ATTRIBUTES (APPC) command 174
EXTRACT ATTRIBUTES (MRO) command 175
EXTRACT CERTIFICATE command 177
EXTRACT LOGONMSG command 179
EXTRACT PROCESS command 180
EXTRACT TCPIP command 182
EXTRACT TCT command 185
EXTRACT WEB command 186

F

FACILITY option
ASSIGN command 48
FACILITYTOKN option
RUN command 477
FCI option
ASSIGN command 49, 58
FCT option
DUMP TRANSACTION command 154
FETCH ANY command 191
FETCH CHILD command 192
field
extracting information 664
field definition macro, BMS 855
FIELD option
BIF DEEDIT command 59
field separator operand 868, 877
FIELD value
DFHMDI 868
DFHMSD 877
FIELDS operand
DFHMDI 868
file control
commands 20
deleting VSAM records 120
end browse operation 158
read next record 385
read previous record 395
release exclusive control 606
specify start for browse 580
update a record 465
writing new record 715
FILE option
DELETE command 120
ENDBR command 159
READ command 375
READNEXT command 388
READPREV command 399
RESETBR command 448
REWRITE command 466
STARTBR command 582
UNLOCK command 608
WRITE command 716
filename
definition 4, 5, 6, 8, 9
filename argument, CICS command format 3
FILENOTFOUND condition
DELETE command 120
ENDBR command 159
READ command 380
READNEXT command 392
READPREV command 402
RESETBR command 450
REWRITE command 468
STARTBR command 584
UNLOCK command 608
WRITE command 718
FIRESTATUS option
GETNEXT EVENT command 256
INQUIRE EVENT command 274
TEST EVENT command 597
FIRST value
DFHMDI 868
FLDSEP operand
DFHMDI 868
DFHMSD 877
FLENGTH option
DUMP TRANSACTION command 154
fullword alternative to LENGTH 841
GDS RECEIVE command 227
GDS SEND command 229
GET CONTAINER (BTS) command 232
GET CONTAINER (CHANNEL) command 236
GET64 CONTAINER command 259
FLENGTH option *(continued)*
GETMAIN command 246
GETMAIN64 command 250
LOAD command 339
PUT CONTAINER (BTS) command 355
PUT CONTAINER (CHANNEL) command 359
PUT64 CONTAINER command 363
RECEIVE (non-z/OS Communications Server) command 427
RECEIVE (z/OS Communications Server) command 419
SEND (non-z/OS Communications Server) command 497
SEND (z/OS Communications Server) command 491
SIGNAL EVENT command 531
SPOOLWRITE command 559
WRITE JOURNALNAME command 723
FMH option
CONVERSE (non-z/OS Communications Server) command 98
CONVERSE (z/OS Communications Server) command 93
SEND (non-z/OS Communications Server) command 497
SEND (z/OS Communications Server) command 492
START command 567
FMHPARM option
SEND MAP command 506
SEND PAGE command 515
SEND TEXT command 520
FOLD operand
DFHMSD 877
FOR option
DELAY command 117
FORCE TIMER command 194
form field
extracting information 664
FORMATTIME command 195
FORMFEED option
SEND CONTROL command 501
SEND MAP command 506
SEND MAP MAPPINGDEV command 512
SEND TEXT command 520
FORMFIELD option
WEB READ FORMFIELD command 665
WEB READNEXT FORMFIELD command 671
WEB STARTBROWSE FORMFIELD command 708
FREE (APPC) command 201
FREE (LUTYPE6.1) command 202
FREE (MRO) command 203
FREE CHILD command 204
FREE command 200
free main storage 205, 208
FREEKB option
SEND CONTROL command 501
SEND MAP command 506

FREEKB option *(continued)*
SEND MAP MAPPINGDEV command 512
SEND TEXT command 520
SEND TEXT NOEDIT command 528
FREEKB value
DFHMDI 868
DFHMSD 877
FREEMAIN command 205
FREEMAIN64 command 208
FROM option
CONVERSE (non-z/OS Communications Server) command 99
CONVERSE (z/OS Communications Server) command 93
DUMP TRANSACTION command 154
ENTER TRACENUM command 167
GDS SEND command 229
ISSUE ADD command 290
ISSUE PASS command 308
ISSUE REPLACE command 315
ISSUE SEND command 318
PUT CONTAINER (BTS) command 356
PUT CONTAINER (CHANNEL) command 359
PUT64 CONTAINER command 363
RECEIVE MAP command 431
RECEIVE MAP MAPPINGDEV command 434
REWRITE command 467
SEND (non-z/OS Communications Server) command 497
SEND (z/OS Communications Server) command 492
SEND MAP command 506
SEND MAP MAPPINGDEV command 513
SEND TEXT command 520
SEND TEXT MAPPED command 525
SEND TEXT NOEDIT command 528
SIGNAL EVENT command 531
SPOOLWRITE command 559
START ATTACHcommand 572
START command 567
WEB CONVERSE command 641
WEB SEND command (Client) 703
WEB SEND command (Server) 693
WRITE command 716
WRITE JOURNALNAME command 723
WRITEQ TD command 729
WRITEQ TS command 732
FROMACTIVITY option
MOVE CONTAINER (BTS) command 345
FROMCCSID option
PUT CONTAINER (CHANNEL) command 359
PUT64 CONTAINER command 363
FROMCHANNEL option
SIGNAL EVENT command 531
FROMCODEPAGE option
GET CONTAINER (CHANNEL) command 359, 363
FROMDOC option
DOCUMENT INSERT command 147
FROMFLENGTH option
CONVERSE (non-z/OS Communications Server) command 99
CONVERSE (z/OS Communications Server) command 93
fullword alternative to FROMLENGTH 841
FROMLENGTH option
CONVERSE (non-z/OS Communications Server) command 99
CONVERSE (z/OS Communications Server) command 93
ENTER TRACENUM command 167
fullword length alternative (FROMFLENGTH) 841
WEB CONVERSE command 641
WEB SEND command (Client) 703
WEB SEND command (Server) 694
FROMPROCESS option
MOVE CONTAINER (BTS) command 345
FRSET option
SEND CONTROL command 501
SEND MAP command 506
SEND MAP MAPPINGDEV command 513
FRSET value
DFHMDI 868
DFHMSD 877
FSET value
DFHMDF 857
Full Function Logical Unit, 3790 91, 418, 489
FULLDATE option
FORMATTIME 198
fullword length option 841
FUNCERR condition
ISSUE ABORT command 289
ISSUE ADD command 291
ISSUE END command 298
ISSUE ERASE command 302
ISSUE NOTE command 307
ISSUE QUERY command 312
ISSUE REPLACE command 316
ISSUE SEND command 319
ISSUE WAIT command 323
Function codes
of EXEC CICS commands 774

G

GCHARS option
ASSIGN command 49
GCODES option
ASSIGN command 49
GDS (generalized data stream) 17
GDS ALLOCATE command 210
GDS ASSIGN command 212
GDS CONNECT PROCESS command 213
GDS EXTRACT ATTRIBUTES command 215
GDS EXTRACT PROCESS command 216
GDS FREE command 218
GDS ISSUE ABEND command 219
GDS ISSUE CONFIRMATION command 221
GDS ISSUE ERROR command 222
GDS ISSUE PREPARE command 223
GDS ISSUE SIGNAL command 225
GDS RECEIVE command 226
GDS SEND command 228
GDS WAIT command 230
General Banking Terminal System (2980 General Banking Terminal System) 424
generalized data stream (GDS) 17
generic applid, XRF 44
GENERIC option
DELETE command 120
READ command 375
RESETBR command 448
STARTBR command 582
GET CONTAINER (BTS) command 232
GET CONTAINER (CHANNEL) command 234
GET COUNTER command 239
GET DCOUNTER command 239
get main 64-bit storage 248
get main storage 244
GETMAIN command 244
GETMAIN64 command 248
GETNEXT ACTIVITY command 252
GETNEXT CONTAINER (CHANNEL) command 254
GETNEXT CONTAINER command 253
GETNEXT EVENT command 255
GETNEXT PROCESS command 257
GINIT operand
DFHMDF 857
GMMI option
ASSIGN command 49
GROUPID option
SIGNON command 534
SIGNON TOKEN command 538
GRPNAME operand
DFHMDF 857
GTEQ option
READ command 375
RESETBR command 448
STARTBR command 582

H

HANDLE ABEND command 262
HANDLE AID command 264
HANDLE CONDITION command 267
header
browsing 650, 708
retrieve next 670
HEADER operand
DFHMDI 868
HEADER option
SEND TEXT command 520
hhmmss argument, CICS command format 3

HILIGHT operand
DFHMDF 857
DFHMDI 868
DFHMSD 877
HILIGHT option
ASSIGN command 49
HOLD option
LOAD command 339
HONEOM option
SEND CONTROL command 501
SEND MAP command 506
SEND TEXT command 521
SEND TEXT NOEDIT command 528
host command processor LU, 3650/3680 488
host conversational LU 3650
(3270) 88, 487
(3653) 89, 487
HOST option
WEB EXTRACT or EXTRACT WEB command 187, 653
WEB OPEN command 659
WEB PARSE URL command 663
HOSTCODEPAGE option
WEB READ FORMFIELD command 665
WEB READ QUERYPARM command 669
WEB RECEIVE command (Server) 677
WEB SEND command (Server) 694
WEB STARTBROWSE FORMFIELD command 709
WEB STARTBROWSE QUERYPARM command 711
HOSTLENGTH option
WEB EXTRACT or EXTRACT WEB command 187, 653
WEB OPEN command 659
WEB PARSE URL command 663
HOSTTYPE option
WEB EXTRACT or EXTRACT WEB command 188, 654
WEB PARSE URL command 663
HOURS option
DEFINE TIMER command 116
DELAY command 117
POST command 351
ROUTE command 472
START command 567
HTAB operand
DFHMSD 877
HTTPHEADER option
WEB READ HTTPHEADER command 667
WEB READNEXT HTTPHEADER command 672
WEB WRITE HTTPHEADER command 714
HTTPMETHOD option
WEB EXTRACT or EXTRACT WEB command 188, 654
HTTPRNUM option
WEB OPEN command 659
HTTPVERSION option
WEB EXTRACT or EXTRACT WEB command 188, 654
HTTPVNUM option
WEB OPEN command 659

I

IC value
DFHMDF 857
IGNORE CONDITION command 268
IGREQCD condition
CONVERSE (z/OS Communications Server) command 95
ISSUE SEND command 319
SEND (z/OS Communications Server) command 493
SEND CONTROL command 503
SEND MAP command 509
SEND PAGE command 516
SEND TEXT command 523
SEND TEXT MAPPED command 526
SEND TEXT NOEDIT command 529
IGREQID condition
ROUTE command 474
SEND CONTROL command 503
SEND MAP command 510
SEND TEXT command 523
SEND TEXT MAPPED command 526
SEND TEXT NOEDIT command 529
ILLOGIC condition
DELETE command 120
ENDBR command 159
ENDBROWSE ACTIVITY command 160
ENDBROWSE CONTAINER (CHANNEL) command 162
ENDBROWSE CONTAINER command 161
ENDBROWSE PROCESS command 163
GETNEXT ACTIVITY command 253
GETNEXT CONTAINER (CHANNEL) command 255
GETNEXT CONTAINER command 254
GETNEXT PROCESS command 257
INQUIRE PROCESS command 276
READ command 380
READNEXT command 392
READPREV command 403
RESETBR command 450
REWRITE command 468
SPOOLOPEN INPUT command 550
SPOOLOPEN OUTPUT command 555
SPOOLREAD command 557
STARTBR command 584
UNLOCK command 608
WEB STARTBROWSE FORMFIELD command 709
WEB STARTBROWSE HTTPHEADER command 710
WEB STARTBROWSE QUERYPARM command 712
WRITE command 718
IMMEDIATE option
RETURN command 461
implicit SPOOLCLOSE 550
INBFMH condition
CONVERSE (non-z/OS Communications Server) command 101
CONVERSE (z/OS Communications Server) command 95
RECEIVE (non-z/OS Communications Server) command 429
RECEIVE (z/OS Communications Server) command 421
INCREMENT option
GET COUNTER command 239
GET DCOUNTER command 239
REWIND COUNTER command 463
REWIND DCOUNTER command 463
INITIAL mode, of an activity 270
INITIAL operand
DFHMDF 857
initialize main 64-bit storage 248
initialize main storage 244
initiate a task 561
INITIMG option
GETMAIN command 246
INITPARM option
ASSIGN command 49
INITPARMLEN option
ASSIGN command 49
INPARTN option
ASSIGN command 49
RECEIVE MAP command 431
input operation without data 846
INPUTEVENT option
LINK ACQPROCESS command 333
LINK ACTIVITY command 336
RUN command 477
INPUTMSG option
LINK command 326
RETURN command 461
XCTL command 748
INPUTMSGLEN option
ASSIGN command 49
LINK command 327
RETURN command 461
XCTL command 748
INQUIRE ACTIVITYID command 269
INQUIRE CONTAINER command 272
INQUIRE EVENT command 274
INQUIRE PROCESS command 275
INQUIRE TIMER command 276
interactive logical units 90, 417, 488
interface processor DFHEAI 13
interpreter logical unit, 3650
CONVERSE command 88
ISSUE EODS command 300
ISSUE LOAD command 305
RECEIVE command 417
SEND (z/OS Communications Server) command 486
interrogate a data set 312
interval control
ASKTIME options 40
cancel interval control command 66
CANCEL options 67
commands 21
DELAY options 117

interval control *(continued)*
delay processing of task 117
FORMATTIME options 197
notification when specified time expires 349
request current time of day 40
retrieve data stored for task 453
start a task 561
wait for event to occur 624
INTERVAL option
DELAY command 117
POST command 351
ROUTE command 472
START command 567
INTO option
CONVERSE (non-z/OS Communications Server) command 99
CONVERSE (z/OS Communications Server) command 93
DOCUMENT RETRIEVE command 150
EXTRACT LOGONMSG command 180
GDS RECEIVE command 227
GET CONTAINER (BTS) command 233
GET CONTAINER (CHANNEL) command 237
GET64 CONTAINER command 260
ISSUE RECEIVE command 314
READ command 376
READNEXT command 388
READPREV command 399
READQ TD command 406
READQ TS command 409
RECEIVE (non-z/OS Communications Server) command 427
RECEIVE (z/OS Communications Server) command 419
RECEIVE MAP command 431
RECEIVE MAP MAPPINGDEV command 434
RECEIVE PARTN command 436
RETRIEVE command 454
SPOOLREAD command 557
WEB CONVERSE command 644
WEB RECEIVE command (Client) 684
WEB RECEIVE command (Server) 677
INTOCCSID option
GET CONTAINER (CHANNEL) command 237
GET64 CONTAINER command 260
INTOCODEPAGE option
GET CONTAINER (CHANNEL) command 237
GET64 CONTAINER command 260
INVALIDCOUNT option
VERIFY PASSWORD command 615
VERIFY PHRASE command 618
INVERRTERM condition
ROUTE command 474
INVITE option
GDS SEND command 229
INVITE option *(continued)*
SEND (non-z/OS Communications Server) command 497
SEND (z/OS Communications Server) command 492
INVLDC condition
ROUTE command 474
SEND CONTROL command 503
SEND MAP command 510
SEND TEXT command 524
INVMPSZ condition
EIBRCODE byte 3 755
RECEIVE MAP command 432
RECEIVE MAP MAPPINGDEV command 435
SEND MAP command 510
SEND MAP MAPPINGDEV command 514
INVOKE APPLICATION command 278
INVOKE SERVICE command 281
INVOKE WEBSERVICE command 281, 286
INVOKINGPROG option
ASSIGN command 49
INVPARTN condition
RECEIVE MAP command 432
RECEIVE PARTN command 437
SEND CONTROL command 503
SEND MAP command 510
SEND TEXT command 524
SEND TEXT NOEDIT command 529
INVPARTNSET condition
SEND PARTNSET command 518
INVREQ condition
ACQUIRE command 28
ADD SUBEVENT command 30
ALLOCATE (APPC) command 36
ALLOCATE (LUTYPE6.1) command 38
ALLOCATE (MRO) command 39
ASSIGN command 56
BIF DIGEST command 60
CANCEL (BTS) command 69
CHANGE PASSWORD command 73
CHANGE PHRASE command 71
CHANGE TASK command 74
CHECK ACQPROCESS command 76
CHECK ACTIVITY command 78
CHECK TIMER command 80
CONNECT PROCESS command 82
CONVERSE (z/OS Communications Server) command 95
CONVERTTIME command 103
DEFINE ACTIVITY command 105
DEFINE COMPOSITE EVENT command 108
DEFINE INPUT EVENT command 112
DEFINE PROCESS command 114
DEFINE TIMER command 117
DELAY command 117
DELETE ACTIVITY command 128
DELETE command 120
DELETE CONTAINER (BTS) command 131
DELETE CONTAINER (CHANNEL) command 132
INVREQ condition *(continued)*
DELETE COUNTER command 108, 133, 365
DELETE DCOUNTER command 108
DELETE EVENT command 135
DELETE TIMER command 136
DELETEQ TD command 137
DELETEQ TS command 139
DEQ command 141
DUMP TRANSACTION command 157
EIBRCODE bytes 1-3 755
ENDBR command 159
ENQ command 166
ENTER TRACENUM command 167
EXTRACT ATTACH (LUTYPE6.1) command 171
EXTRACT ATTACH (MRO) command 174
EXTRACT ATTRIBUTES (APPC) command 175
EXTRACT CERTIFICATE command 179
EXTRACT PROCESS command 180
EXTRACT TCPIP command 185
EXTRACT TCT command 186
FORCE TIMER command 195
FORMATTIME command 200
FREE (APPC) command 202
FREE (LUTYPE6.1) command 203
FREE (MRO) command 204
FREEMAIN command 206, 209
GET CONTAINER (BTS) command 233
GET CONTAINER (CHANNEL) command 239
GET64 CONTAINER command 262
GETMAIN64 command 251
HANDLE AID command 266
INQUIRE EVENT command 275
INQUIRE TIMER command 277
INVOKE APPLICATION command 281
ISSUE ABEND command 287
ISSUE ABORT command 289
ISSUE ADD command 291
ISSUE CONFIRMATION command 292
ISSUE END command 298
ISSUE ENDFILE command 299
ISSUE ENDOUTPUT command 299
ISSUE EODS command 300
ISSUE ERASE command 302
ISSUE ERASEAUP command 303
ISSUE ERROR command 305
ISSUE NOTE command 307
ISSUE PASS command 309
ISSUE PREPARE command 310
ISSUE PRINT command 311
ISSUE QUERY command 313
ISSUE RECEIVE command 314
ISSUE REPLACE command 316
ISSUE SEND command 319
ISSUE SIGNAL (APPC) command 320
ISSUE WAIT command 323
LINK ACQPROCESS command 334

INVREQ condition *(continued)*
LINK ACTIVITY command 337
LINK command 329
LOAD command 339
MONITOR command 342
MOVE CONTAINER (BTS) command 345
MOVE CONTAINER (CHANNEL) command 348
POP HANDLE command 349
POST command 352
PURGE MESSAGE command 353
PUT CONTAINER (BTS) command 356
PUT CONTAINER (CHANNEL) command 360
PUT64 CONTAINER command 364
QUERY SECURITY command 371
READ command 380
READNEXT command 392
READPREV command 403
READQ TD command 407
READQ TS command 411
RECEIVE (non-z/OS Communications Server) command 429
RECEIVE MAP command 433
RECEIVE MAP MAPPINGDEV command 435
RECEIVE PARTN command 437
RELEASE command 438
REMOVE SUBEVENT command 440
REQUEST PASSTICKET command 442, 444
RESET ACQPROCESS command 445
RESET ACTIVITY command 446
RESETBR command 450
RESUME command 452
RETRIEVE command 455
RETRIEVE REATTACH EVENT command 457
RETRIEVE SUBEVENT command 459
RETURN command 462
REWRITE command 468
ROUTE command 474
RUN command 478
SEND (non-z/OS Communications Server) command 498
SEND CONTROL command 503
SEND MAP command 510
SEND MAP MAPPINGDEV command 514
SEND PAGE command 516
SEND PARTNSET command 518
SEND TEXT command 524
SEND TEXT MAPPED command 526
SEND TEXT NOEDIT command 530
SIGNOFF command 532
SIGNON command 535, 539
SOAPFAULT ADD command 542
SOAPFAULT CREATE command 546
SOAPFAULT DELETE command 547
SPOOLCLOSE command 548
SPOOLOPEN INPUT command 550
SPOOLOPEN OUTPUT command 555
INVREQ condition *(continued)*
SPOOLREAD command 558
SPOOLWRITE command 560
START ATTACH command 572
START BREXITcommand 573
START command 561
START TRANSID (CHANNEL) command 576
STARTBR command 584
STARTBROWSE EVENT command 591
SUSPEND (BTS) command 594
SYNCPOINT command 595
SYNCPOINT ROLLBACK command 597
TEST EVENT command 597
UNLOCK command 608
VERIFY PASSWORD command 615, 618
VERIFY TOKEN command 622
WAIT CONVID command 624
WAIT EVENT command 625
WAIT EXTERNAL command 627
WAIT TERMINAL command 632
WAITCICS command 634
WEB CONVERSE command 648
WEB ENDBROWSE FORMFIELD command 650
WEB ENDBROWSE HTTPHEADER command 651
WEB ENDBROWSE QUERYPARM command 652
WEB EXTRACT or EXTRACT WEB command 190, 656
WEB OPEN command 661
WEB PARSE URL command 664
WEB READ FORMFIELD command 666
WEB READ HTTPHEADER command 668
WEB READ QUERYPARM command 670
WEB READNEXT FORMFIELD command 671
WEB READNEXT HTTPHEADER command 672
WEB READNEXT QUERYPARM command 674
WEB RECEIVE command (Client) 687
WEB RECEIVE command (Server) 681
WEB RETRIEVE command 689
WEB SEND command (Client) 706
WEB SEND command (Server) 696
WEB STARTBROWSE FORMFIELD command 709
WEB STARTBROWSE HTTPHEADER command 710
WEB STARTBROWSE QUERYPARM command 712
WEB WRITE HTTPHEADER command 714
WRITE command 719
WRITE JOURNALNAME command 724
WRITE OPERATOR command 728
INVREQ condition *(continued)*
WRITEQ TD command 729
WRITEQ TS command 733
WSACONTEXT BUILD command 739
XCTL command 749
INVREQ option
DOCUMENT RETRIEVE command 150
IOERR condition
ACQUIRE command 28
CANCEL (BTS) command 69
CHECK ACTIVITY command 79
CHECK TIMER command 80
DEFINE ACTIVITY command 105
DEFINE PROCESS command 114
DELETE ACTIVITY command 128
DELETE command 120
DELETE CONTAINER (BTS) command 131
DUMP TRANSACTION command 157
ENDBR command 159
GET CONTAINER (BTS) command 234
GETNEXT ACTIVITY command 253
GETNEXT PROCESS command 257
INQUIRE CONTAINER command 273
INQUIRE EVENT command 275
INQUIRE TIMER command 277
LINK ACQPROCESS command 334
LINK ACTIVITY command 337
MOVE CONTAINER (BTS) command 345
PUT CONTAINER (BTS) command 356
READ command 381
READNEXT command 393
READPREV command 403
READQ TD command 407
READQ TS command 411
RESET ACQPROCESS command 445
RESET ACTIVITY command 447
RESETBR command 450
RESUME command 452
RETRIEVE command 455
REWRITE command 468
RUN command 478
START command 561
STARTBR command 585
STARTBROWSE CONTAINER command 589
STARTBROWSE EVENT command 592
STARTBROWSE PROCESS command 592
SUSPEND (BTS) command 594
UNLOCK command 609
WEB CONVERSE command 650
WEB EXTRACT or EXTRACT WEB command (Client) 190, 656
WEB OPEN command 661
WEB RECEIVE command (Client) 688
WEB SEND command (Client) 707
WEB SEND command (Server) 697

IOERR condition *(continued)*
WRITE command 719
WRITE JOURNALNAME command 724
WRITEQ TD command 730
WRITEQ TS command 733
IOERR option
WAIT JOURNALNAME command 630
ISCINVREQ condition
CANCEL command 68
DELETE command 120
DELETEQ TD command 137
DELETEQ TS command 139
ENDBR command 160
READ command 381
READNEXT command 393
READPREV command 404
READQ TD command 407
READQ TS command 411
RESETBR command 451
REWRITE command 469
START command 561
START TRANSID (CHANNEL) command 576
STARTBR command 585
UNLOCK command 609
WRITE command 719
WRITEQ TD command 730
WRITEQ TS command 734
ISSUE ABEND command 287
ISSUE ABORT command 288
ISSUE ADD command 290
ISSUE CONFIRMATION command 292
ISSUE COPY (3270 logical) command 293
ISSUE COPY command
general information 845
ISSUE DISCONNECT (default) command 294
ISSUE DISCONNECT (LUTYPE6.1) command 296
ISSUE DISCONNECT command
general information 841
ISSUE END command 297
ISSUE ENDFILE command 298
ISSUE ENDOUTPUT command 299
ISSUE EODS command 300
ISSUE ERASE command 301
ISSUE ERASEAUP command 303
general information 845
ISSUE ERROR command 304
ISSUE LOAD command 305
ISSUE NOTE command 306
ISSUE PASS command 308
ISSUE PREPARE command 309
ISSUE PRINT command 311
general information 845
ISSUE QUERY command 312
ISSUE RECEIVE command 313
ISSUE REPLACE command 315
ISSUE RESET command 317
ISSUE SEND command 317
ISSUE SIGNAL (APPC) command 319
ISSUE SIGNAL (LUTYPE6.1) command 321
ISSUE SIGNAL command
general information 841
ISSUE WAIT command 322
ISSUER option
EXTRACT CERTIFICATE command 178
ISUSERID option
VERIFY TOKEN command 622
ITEM option
READQ TS command 409
WRITEQ TS command 732
ITEMERR condition
READQ TS command 411
WRITEQ TS command 734
IUTYPE option
BUILD ATTACH (LUTYPE6.1) command 62
BUILD ATTACH (MRO) command 65
EXTRACT ATTACH (LUTYPE6.1) command 169
EXTRACT ATTACH (MRO) command 172

J

JIDERR condition
WRITE JOURNALNAME command 724
JIDERR option
WAIT JOURNALNAME command 630
JOURNAL command 323
journal control
create a journal record 323
journal record, creating 722
journaling commands 21
JOURNALNAME option
WAIT JOURNALNAME command 629
WRITE JOURNALNAME command 723
JTYPEID option
WRITE JOURNALNAME command 723
JUSFIRST option
SEND TEXT command 521
JUSLAST option
SEND TEXT command 521
JUSTIFY operand
DFHMDF 857
DFHMDI 868
JUSTIFY option
SEND TEXT command 521

K

katakana and English characters, mixed 98, 436
KATAKANA option
ASSIGN command 50
katakana terminals
CONVERSE (3270 logical) command 92
CONVERSE (LUTYPE2/LUTYPE3) command 92
katakana terminals *(continued)*
CONVERSE command (3270 display) 98
CONVERSE command (3600 BTAM) 98
CONVERSE command (3735) 98
CONVERSE command (3740) 98
CONVERSE command (System/3) 98
CONVERSE command (System/7) 98
RECEIVE (non-z/OS Communications Server) command 427
RECEIVE (z/OS Communications Server) command 419
RECEIVE MAP command 431
RECEIVE PARTN command 436
SEND (non-z/OS Communications Server) command 496
KEEP option
SPOOLCLOSE command 548
KEYLENGTH option
DELETE command 120
ISSUE ERASE command 301
ISSUE REPLACE command 316
READ command 376
READNEXT command 388
READPREV command 399
RESETBR command 448
STARTBR command 583
WRITE command 716
KEYNUMBER option
ISSUE ERASE command 301
ISSUE REPLACE command 316
keyword length 841

L

L40, L64, or L80 options
SEND CONTROL command 501
SEND MAP command 507
SEND TEXT command 521
SEND TEXT NOEDIT command 528
label argument, CICS command format 3
LABEL option
HANDLE ABEND command 263
LANG operand
DFHMSD 877
LANGINUSE option
ASSIGN 50
SIGNON command 534
SIGNON TOKEN command 539
language codes 839
LANGUAGECODE option
SIGNON command 534
SIGNON TOKEN command 538
large COMMAREAs, channels 129, 132, 234, 345, 357, 365, 459, 576, 747
LAST option
GDS SEND command 229
SEND (non-z/OS Communications Server) command 497
SEND (z/OS Communications Server) command 492
SEND CONTROL command 501
SEND MAP command 507

LAST option *(continued)*
SEND PAGE command 515
SEND TEXT command 521
SEND TEXT MAPPED command 525
SEND TEXT NOEDIT command 528
LAST value
DFHMDI 868
LASTUSETIME option
VERIFY PASSWORD command 615
VERIFY PHRASE command 618
LDC operand
DFHMSD 877
LDC option
CONVERSE (z/OS Communications Server) command 93
ROUTE command 472
SEND (z/OS Communications Server) command 492
SEND CONTROL command 501
SEND MAP command 507
SEND TEXT command 521
LDCMNEM option
ASSIGN command 50
LDCNUM option
ASSIGN command 50
LEAVEKB option
CONVERSE (non-z/OS Communications Server) command 99
RECEIVE (non-z/OS Communications Server) command 427
SEND (non-z/OS Communications Server) command 497
LEFT value
DFHMDF 857
DFHMDI 868
LENGERR condition
BIF DEEDIT command 59
BIF DIGEST command 60
CHANGE PHRASE command 72
CONNECT PROCESS command 82
CONVERSE (non-z/OS Communications Server) command 101
CONVERSE (z/OS Communications Server) command 96
DEQ command 141
EIBRCODE byte 1 755
ENQ command 166
ENTER TRACENUM command 168
EXTRACT CERTIFICATE command 179
EXTRACT PROCESS command 180
EXTRACT TCPIP command 185
GET CONTAINER (BTS) command 234
GET CONTAINER (CHANNEL) command 239
GET64 CONTAINER command 262
GETMAIN command 247
GETMAIN64 command 251
INVOKE APPLICATION command 281
ISSUE COPY (3270 logical) command 294
ISSUE PASS command 309
LENGERR condition *(continued)*
ISSUE RECEIVE command 315
LINK command 329
LOAD command 340
PUT CONTAINER (CHANNEL) command 361
PUT64 CONTAINER command 364
QUERY SECURITY command 371
READ command 381
READNEXT command 394
READPREV command 404
READQ TD command 407
READQ TS command 411
RECEIVE (non-z/OS Communications Server) command 429
RECEIVE (z/OS Communications Server) command 421
RECEIVE PARTN command 438
RETRIEVE command 455
RETURN command 463
REWRITE command 469
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 494
SEND TEXT command 524
SIGNAL EVENT command 531
SIGNON command 536, 540
SOAPFAULT CREATE command 543, 546
SPOOLOPEN OUTPUT command 555
SPOOLREAD command 558
SPOOLWRITE command 560
START ATTACH command 572
START BREXITcommand 573
START command 561
VERIFY PHRASE command 619
VERIFY Token command 622
WEB CONVERSE command 649
WEB EXTRACT or EXTRACT WEB command 190, 656
WEB OPEN command 662
WEB PARSE URL command 664
WEB READ FORMFIELD command 666
WEB READ HTTPHEADER command 668
WEB READ QUERYPARM command 670
WEB READNEXT FORMFIELD command 671
WEB READNEXT HTTPHEADER command 673
WEB READNEXT QUERYPARM command 674
WEB RECEIVE command (Client) 687
WEB RECEIVE command (Server) 681
WEB SEND command (Client) 697, 707
WEB STARTBROWSE FORMFIELD command 709
WEB STARTBROWSE QUERYPARM command 712
LENGERR condition *(continued)*
WEB WRITE HTTPHEADER command 715
WRITE command 720
WRITE JOURNALNAME command 724
WRITE OPERATOR command 728
WRITEQ TD command 730
WRITEQ TS command 734
XCTL command 749
LENGERR option
DOCUMENT RETRIEVE command 151
LENGTH operand
DFHMDF 857
LENGTH option
BIF DEEDIT command 59
built-in function 59
default (assembler language) 8
default (C) 5
default (PL/I) 7
DEQ command 140
DOCUMENT RETRIEVE command 150
DOCUMENT SET command 152
DUMP TRANSACTION command 155
ENQ command 164
EXTRACT CERTIFICATE command 178
EXTRACT LOGONMSG command 180
fullword length alternative (FLENGTH) 841
GETMAIN command 247
INVOKE APPLICATION command 279
ISSUE ADD command 290
ISSUE PASS command 308
ISSUE RECEIVE command 314
ISSUE REPLACE command 316
ISSUE SEND command 318
LINK command 327
LOAD command 339
READ command 376
READNEXT command 389
READPREV command 399
READQ TD command 406
READQ TS command 409
RECEIVE (non-z/OS Communications Server) command 427
RECEIVE (z/OS Communications Server) command 420
RECEIVE MAP command 431
RECEIVE MAP MAPPINGDEV command 434
RECEIVE PARTN command 437
RETRIEVE command 454
RETURN command 461
REWRITE command 467
SEND (non-z/OS Communications Server) command 497
SEND (z/OS Communications Server) command 492
SEND MAP command 507
SEND MAP MAPPINGDEV command 513

LENGTH option *(continued)*
SEND TEXT command 521
SEND TEXT MAPPED command 525
SEND TEXT NOEDIT command 528
START ATTACHcommand 572
START command 567
WEB RECEIVE command (Client) 684
WEB RECEIVE command (Server) 677
WEB SEND command (Server) 694
WRITE command 717
WRITEQ TD command 729
WRITEQ TS command 732
XCTL command 748
LENGTH value
DFHMDI 868
DFHMSD 877
LENGTHLIST option
DUMP TRANSACTION command 155
LEVEL option
GETNEXT ACTIVITY command 252
LIGHTPEN option
HANDLE AID command 264
LINE operand
DFHMDI 868
LINE option
SPOOLWRITE command 559
line value
DFHMDI 868
line,column value
DFHMDF 857
LINEADDR option
CONVERSE (non-z/OS Communications Server) command 99
SEND (non-z/OS Communications Server) command 497
LINK ACQPROCESS command 332
LINK ACTIVITY command 335
LINK command 323
link to program expecting return 323
LINKLEVEL option
ASSIGN command 50
LIST option
ROUTE command 473
literal constants 7
LLID option
GDS RECEIVE command 227
LOAD command 338
load programs, tables, or maps 338
LOADING condition
DELETE command 120
READ command 382
READNEXT command 394
STARTBR command 585
WRITE command 720
LOCAL 199
LOCALITY option
EXTRACT CERTIFICATE command 178
LOCALITYLEN option
EXTRACT CERTIFICATE command 178
LOCATION(LOC24) option
GETMAIN64 command 250
LOCKED condition
ACQUIRE command 28
CANCEL (BTS) command 69
CHECK ACTIVITY command 79
DELETE ACTIVITY command 129
DELETE command 120
DELETE CONTAINER (BTS) command 131
DELETEQ TD command 137
DELETEQ TS command 139
GET CONTAINER (BTS) command 234
LINK ACTIVITY command 337
MOVE CONTAINER (BTS) command 345
PUT CONTAINER (BTS) command 356
READ command 382
READNEXT command 394
READPREV command 404
READQ TD command 408
RESET ACQPROCESS command 445
RESET ACTIVITY command 447
RESUME command 453
REWRITE command 469
RUN command 478
SUSPEND (BTS) command 594
WRITE command 720
WRITEQ TD command 730
WRITEQ TS command 734
logical device code (LDC) 87, 485
logical messages, BMS
completing a logical message 514
full BMS
ROUTE command 471
purging a logical message 353
routing a logical message 471
LOGMESSAGE option
QUERY SECURITY command 369
LOGMODE option
ISSUE PASS command 308
LOGONLOGMODE option
ISSUE PASS 309
LU (logical unit)
3270 Information Display System 86, 293, 415, 484
3270 SCS Printer 85, 483
3270-Display, LUTYPE2 83, 413, 482
3270-Display, LUTYPE3 413, 482
3600 (3601) 86, 416, 485
3600 (3614) 87, 416, 486
3600 pipeline 415, 484
3650 host conversational (3270) 88, 487
3650 host conversational (3653) 89, 487
3650 interpreter 88, 300, 305, 417, 486
3650/3680 host command processor 488
3770 batch 90, 418, 488
3790 (3270-display) 91, 426, 490
3790 (3270-printer) 490
3790 full-function 418, 489
3790 full-function or inquiry 91
3790 SCS printer 489
LU (logical unit) *(continued)*
batch 90, 418, 488
conversing with (CONVERSE) 841
interactive 90
reading data from 313, 841
writing data to 290, 841
LUNAME option
ISSUE PASS command 309
LUTYPE2, 3270-Display LU 83, 413, 482
LUTYPE3, 3270-Display LU 413, 482
LUTYPE4
logical unit 84, 414, 482
LUTYPE6.1 logical unit
acquiring a session 37
communicating on LUTYPE6.1 session 84
converting 8-character names to 4 characters 185
disconnecting 296
getting information about 348
receiving data 414
requesting change of direction 321
retrieving values from an LUTYPE6.1 header 168
sending data 483
specifying values for an MRO attach header 64
specifying values for LUTYPE6.1 attach header 61

M

macros, BMS, summary 855
magnetic slot reader (MSR) 852
main 64-bit storage 248
MAIN option
WRITEQ TS command 732
main storage 244
MAJORVERSION option 50
INVOKE APPLICATION command 280
map definition macro, BMS 855, 868
MAP option
RECEIVE MAP command 432
RECEIVE MAP MAPPINGDEV command 435
SEND MAP command 507
SEND MAP MAPPINGDEV command 513
MAP value
DFHMSD 877
MAPATTS operand
DFHMDI 868
DFHMSD 877
MAPCOLUMN option
ASSIGN command 50
MAPFAIL condition
RECEIVE MAP command 433
RECEIVE MAP MAPPINGDEV command 435
MAPHEIGHT option
ASSIGN command 50
MAPLINE option
ASSIGN command 50
MAPONLY option
SEND MAP command 507

MAPONLY option *(continued)*
SEND MAP MAPPINGDEV command 513
MAPONLY value
DFHMDI 868
DFHMSD 877
MAPPINGDEV option
RECEIVE MAP MAPPINGDEV command 435
SEND MAP MAPPINGDEV command 513
maps, loading 338
mapset definition macro (DFHMSD) 855, 877
MAPSET option
RECEIVE MAP command 432
RECEIVE MAP MAPPINGDEV command 435
SEND MAP command 508
SEND MAP MAPPINGDEV command 513
MAPSFX operand
DFHPDI 888
MAPWIDTH option
ASSIGN command 51
MASSINSERT option
WRITE command 717
MAXDATALEN option
EXTRACT TCPIP command 184
MAXFLENGTH option
CONVERSE (non-z/OS Communications Server) command 99
CONVERSE (z/OS Communications Server) command 93
fullword alternative to MAXLENGTH 841
GDS RECEIVE command 227
RECEIVE (non-z/OS Communications Server) command 428
RECEIVE (z/OS Communications Server) command 420
SPOOLREAD command 557
MAXIMUM option
DEFINE COUNTER command 108
DEFINE DCOUNTER command 108
QUERY COUNTER command 365
QUERY DCOUNTER command 365
MAXLENGTH option
CONVERSE (non-z/OS Communications Server) command 99
CONVERSE (z/OS Communications Server) command 93
DOCUMENT RETRIEVE command 150
fullword length alternative (MAXFLENGTH) 841
RECEIVE (non-z/OS Communications Server) command 428
RECEIVE (z/OS Communications Server) command 420
WEB CONVERSE command 644
WEB RECEIVE command (Client) 684
WEB RECEIVE command (Server) 678
MAXLENGTH option *(continued)*
WRITE OPERATOR command 727
MAXLIFETIME option
DEQ command 141
ENQ command 165
MAXPROCLEN option
EXTRACT PROCESS command 180
GDS EXTRACT PROCESS command 217
MCC option
SPOOLOPEN OUTPUT command 553
MEDIATYPE option
WEB CONVERSE command 641, 644
WEB RECEIVE command (Client) 685
WEB RECEIVE command (Server) 678
WEB SEND command (Client) 703
WEB SEND command (Server) 694
METHOD option
WEB CONVERSE command 642
WEB SEND command (Client) 704
METHODLENGTH option
WEB EXTRACT or EXTRACT WEB command 188, 654
MICROVERSION option 51
MILLISECONDS option
FORMATTIME command 198
MINIMUM option
DEFINE COUNTER command 108
DEFINE DCOUNTER command 108
INVOKE APPLICATION command 280
QUERY COUNTER command 365
QUERY DCOUNTER command 365
MINORVERSION option 51
INVOKE APPLICATION command 280
MINUTES option
DEFINE TIMER command 116
DELAY command 117
POST command 351
ROUTE command 473
START command 568
MMDDYY option
FORMATTIME command 198
MMDDYYYY option
FORMATTIME command 198
MODE operand
DFHMSD 877
MODE option
CHECK ACQPROCESS command 75
CHECK ACTIVITY command 78
INQUIRE ACTIVITYID command 270
model codes (terminal) 57
MODENAME option
GDS ALLOCATE command 210
modes, of an activity
ACTIVE 270
CANCELLING 270
COMPLETE 270
DORMANT 270
INITIAL 270
MONITOR command 341
monitoring application performance 341
monitoring commands 21
MONTH option
DEFINE TIMER command 116
MONTHOFYEAR option
FORMATTIME command 198
MOVE CONTAINER (BTS) command 343
MOVE CONTAINER (CHANNEL) command 345
MSR (magnetic slot reader)
control byte values and constants 852
DFHMSRCA, 852
MSR option
SEND CONTROL command 502
SEND MAP command 508
SEND TEXT command 522
MSRCONTROL option
ASSIGN command 51
multi region operation (MRO) commands
ALLOCATE 38
BUILD ATTACH 64
CONVERSE command 97
EXTRACT ATTACH 171
EXTRACT ATTRIBUTES 175
FREE 203
RECEIVE 422
SEND 495
MUSTENTER value
DFHMDF 857
DFHMDI 868
DFHMSD 877
MUSTFILL value
DFHMDF 857
DFHMDI 868
DFHMSD 877

N

name argument, CICS command format 3
NAME option
WAIT EVENT command 625
WAIT EXTERNAL command 627
WAITCICS command 634
named counter
define named counter 108
delete named counter 133
query named counter 365
named counter server commands 21
named counter server, GET command 239
named counter server, REWIND command 463
named counter server, UPDATE command 610
NAMELENGTH option
WEB READ FORMFIELD command 666
WEB READ HTTPHEADER command 667
WEB READ QUERYPARM command 669
WEB READNEXT FORMFIELD command 671
WEB READNEXT HTTPHEADER command 672

NAMELENGTH option *(continued)*
WEB READNEXT QUERYPARM command 673
WEB STARTBROWSE FORMFIELD command 709
WEB STARTBROWSE QUERYPARM command 712
WEB WRITE HTTPHEADER command 714
national language codes 839
NATLANG option
SIGNON command 534
SIGNON TOKEN command 539
NATLANGINUSE option
ASSIGN command 51
SIGNON command 534
SIGNON TOKEN command 539
NETNAME option
ASSIGN command 51
EXTRACT TCT command 186
NETNAMEIDERR condition
ALLOCATE (APPC) command 36
new tasks, passing data to 561
NEWPASSWORD option
CHANGE PASSWORD command 73
SIGNON command 535
NEWPHRASE option
CHANGE PHRASE command 71
SIGNON command 535
NEWPHRASELEN option
CHANGE PHRASE command 71
SIGNON command 535
NEXT option
READQ TS command 410
NEXT value
DFHMDI 868
NEXTTRANSID option
ASSIGN command 51
NLEOM option
ROUTE command 473
SEND MAP command 508
SEND TEXT command 522
NO value
DFHMDI 868
DFHMSD 877
NOAUTOPAGE option
SEND PAGE command 515
NOCC option
SPOOLOPEN OUTPUT command 553
NOCHECK option
DEFINE PROCESS command 112
START command 568
NODATA option
GET CONTAINER (BTS) command 233
GET CONTAINER (CHANNEL) command 237
GET64 CONTAINER command 260
NODE option
SPOOLOPEN OUTPUT command 553
NODEIDERR condition
SPOOLOPEN OUTPUT command 555
NODUMP option
ABEND command 25
NOFLUSH option
SEND MAP command 508
NOHANDLE option
deactivating HANDLE CONDITION command 267
option 10
overriding HANDLE AID 11
NOJBUFSP condition
WRITE JOURNALNAME command 725
NONVAL condition
ISSUE LOAD command 306
NOPASSBKRD condition
RECEIVE (non-z/OS Communications Server) command 430
NOPASSBKWR condition
SEND (non-z/OS Communications Server) command 498
NOQUEUE option
ALLOCATE (APPC) command 35
ALLOCATE (LUTYPE6.1) command 37
ALLOCATE (MRO) command 39
GDS ALLOCATE command 210
NOQUIESCE
ISSUE PASS command 309
NORM value
DFHMDF 857
NOSPACE condition
DUMP TRANSACTION command 157
REWRITE command 469
WRITE command 720
WRITEQ TD command 730
WRITEQ TS command 734
NOSPOOL condition
SPOOLCLOSE command 549
SPOOLOPEN INPUT command 551
SPOOLOPEN OUTPUT command 555
SPOOLREAD command 558
SPOOLWRITE command 560
NOSTART condition
ISSUE LOAD command 306
NOSTG condition
DUMP TRANSACTION command 157
GETMAIN command 247
GETMAIN64 command 251
SPOOLCLOSE command 549
SPOOLOPEN INPUT command 551
SPOOLOPEN OUTPUT command 556
SPOOLREAD command 558
SPOOLWRITE command 560
NOSUSPEND option
ALLOCATE (APPC) 34
ALLOCATE (LUTYPE6.1) command 37
DEFINE COUNTER command 108
DEFINE DCOUNTER command 108
DELETE 120
DELETE COUNTER command 133
DELETE DCOUNTER command 133
ENQ command 165
FETCH CHILD command 192, 193
GET COUNTER command 239
NOSUSPEND option *(continued)*
GET DCOUNTER command 239
GETMAIN command 247
GETMAIN64 command 251
QUERY COUNTER command 365
READ command 377
READNEXT command 390
READPREV command 400
READQ TD command 406
REWIND COUNTER command 463
REWIND DCOUNTER command 463
REWRITE command 467
UPDATE COUNTER command 610
UPDATE DCOUNTER command 610
WRITE command 717
WRITE JOURNALNAME command 724
WRITEQ TS command 732
NOTALLOC condition
CONNECT PROCESS command 82
CONVERSE (non-z/OS Communications Server) command 101
CONVERSE (z/OS Communications Server) command 96
EXTRACT ATTACH (LUTYPE6.1) command 171
EXTRACT ATTACH (MRO) command 174
EXTRACT ATTRIBUTES (APPC) command 175
EXTRACT ATTRIBUTES (MRO) command 176
EXTRACT LOGONMSG command 180
EXTRACT PROCESS command 180
EXTRACT TCT command 186
FREE (APPC) command 202
FREE (LUTYPE6.1) command 203
FREE (MRO) command 204
FREE command 201
ISSUE ABEND command 288
ISSUE CONFIRMATION command 293
ISSUE COPY (3270 logical) command 294
ISSUE DISCONNECT (LUTYPE6.1) command 296
ISSUE ENDFILE command 299
ISSUE ENDOUTPUT command 299
ISSUE EODS command 300
ISSUE ERASEAUP command 303
ISSUE ERROR command 305
ISSUE LOAD command 306
ISSUE PASS command 309
ISSUE PREPARE command 310
ISSUE PRINT command 311
ISSUE SIGNAL (APPC) command 320
ISSUE SIGNAL (LUTYPE6.1) command 321
POINT command 348
RECEIVE (non-z/OS Communications Server) command 430
RECEIVE (z/OS Communications Server) command 422

NOTALLOC condition *(continued)*
SEND (non-z/OS Communications Server) command 499
SEND (z/OS Communications Server) command 494
WAIT CONVID command 624
WAIT SIGNAL command 631
WAIT TERMINAL command 632
NOTAUTH condition
ACQUIRE command 28
CANCEL (BTS) command 69
CANCEL command 68
CHANGE PASSWORD command 72, 73
DEFINE ACTIVITY command 106
DEFINE PROCESS command 114
DELETE command 120
DELETEQ TD command 137
DELETEQ TS command 139
ENDBR command 160
HANDLE ABEND command 264
INQUIRE ACTIVITYID command 271
INQUIRE CONTAINER command 273
INQUIRE EVENT command 275
INQUIRE PROCESS command 276
INQUIRE TIMER command 277
INVOKE APPLICATION command 281
LINK ACQPROCESS command 334
LINK ACTIVITY command 337
LINK command 330
LOAD command 340
READ command 382
READNEXT command 394
READPREV command 404
READQ TD command 408
READQ TS command 411
RELEASE command 439
RESET ACQPROCESS command 445
RESET ACTIVITY command 447
RESETBR command 451
REWRITE command 470
RUN command 478
SIGNON command 536, 540
SPOOLOPEN INPUT command 551
START ATTACH command 572
START BREXIT command 573
START command 561
START TRANSID (CHANNEL) command 576
STARTBR command 586
STARTBROWSE ACTIVITY command 588
STARTBROWSE CONTAINER command 589
STARTBROWSE EVENT command 592
STARTBROWSE PROCESS command 593
UNLOCK command 609
VERIFY PASSWORD command 615
VERIFY PHRASE command 619
VERIFY Token command 623
WEB CONVERSE command 650
WEB OPEN command 662
NOTAUTH condition *(continued)*
WEB SEND command (Client) 707
WRITE command 721
WRITE JOURNALNAME command 725
WRITEQ TD command 730
WRITEQ TS command 734
XCTL command 749
NOTFND condition
CANCEL command 68
DELETE command 120
DELETE COUNTER command 239, 463, 610
QUERY SECURITY command 372
READ command 383
READNEXT command 395
READPREV command 404
RESETBR command 451
REWRITE 469
SOAPFAULT DELETE command 547
SPOOLCLOSE command 549
SPOOLOPEN INPUT command 551
SPOOLOPEN OUTPUT command 556
SPOOLREAD command 558
STARTBR command 586
WEB CONVERSE command 650
WEB EXTRACT or EXTRACT WEB command 191, 657
WEB OPEN command 662
WEB READ FORMFIELD command 667
WEB READ HTTPHEADER command 668
WEB READ QUERYPARM command 670
WEB RECEIVE command (Server) 681
WEB RETRIEVE command 689
WEB SEND command (Client) 707
WEB SEND command (Server) 697
WEB STARTBROWSE FORMFIELD command 709
WEB STARTBROWSE HTTPHEADER command 710
WEB STARTBROWSE QUERYPARM command 712
NOTFND option
DOCUMENT RETRIEVE command 151
NOTOPEN condition
DELETE command 120
DUMP TRANSACTION command 157
READ command 383
READQ TD command 408
SPOOLCLOSE command 549
SPOOLOPEN INPUT command 551
SPOOLOPEN OUTPUT command 556
SPOOLREAD command 558
SPOOLWRITE command 561
STARTBR command 586
UNLOCK command 609
WEB CLOSE command 637
WEB CONVERSE command 648
NOTOPEN condition *(continued)*
WEB ENDBROWSE HTTPHEADER command 651
WEB EXTRACT or EXTRACT WEB command 191, 657
WEB READ HTTPHEADER command 668
WEB READNEXT HTTPHEADER command 673
WEB RECEIVE command (Client) 687
WEB SEND command (Client) 708
WEB STARTBROWSE HTTPHEADER command 710
WEB WRITE HTTPHEADER command 715
WRITE command 721
WRITE JOURNALNAME command 725
WRITEQ TD command 730
NOTOPEN option
WAIT JOURNALNAME command 630
NOTRUNCATE option
CONVERSE (non-z/OS Communications Server) command 99
CONVERSE (z/OS Communications Server) command 93
RECEIVE (non-z/OS Communications Server) command 428
RECEIVE (z/OS Communications Server) command 420
WEB CONVERSE command 645
WEB RECEIVE command (Client) 685
WEB RECEIVE command (Server) 678
NOWAIT option
ISSUE ADD command 291
ISSUE ERASE command 301
ISSUE REPLACE command 316
ISSUE SEND command 318
NSCONTAINER option
TRANSFORM XMLTODATA command 604
NUM value
DFHMDF 857
number value
DFHMDF 857
NUMBER value
DFHMDI 868
NUMCIPHERS option
WEB OPEN command 660
NUMEVENTS option
WAIT EXTERNAL command 627
WAITCICS command 634
NUMITEMS option
READQ TS command 410
WRITEQ TS command 733
NUMREC option
DELETE command 120
ISSUE ADD command 291
ISSUE ERASE command 302
ISSUE REPLACE command 316
NUMROUTES option
WRITE OPERATOR command 727

NUMSEGMENTS option
DUMP TRANSACTION command 155
NUMTAB option
ASSIGN command 51

O

OBFMT operand
DFHMDI 868
DFHMSD 877
OCCURS operand
DFHMDF 857
OFF value
DFHMDF 857
DFHMDI 868
DFHMSD 877
OIDCARD option
SIGNON command 535
ON option
DEFINE TIMER command 116
OPCLASS option
ASSIGN command 51
ROUTE command 473
OPENERR condition
DUMP TRANSACTION command 157
SPOOLOPEN INPUT command 551
SPOOLOPEN OUTPUT command 556
OPERATION option 52
INVOKE APPLICATION command 280
OPERID option
HANDLE AID command 264
OPERKEYS option
ASSIGN command 52
OPERPURGE option
SEND PAGE command 515
OPID option
ASSIGN command 52
OPSECURITY option
ASSIGN command 52
options
BMS 431, 434, 519
length 841
OPTIONS(MAIN)
in PL/I 12
OR option
DEFINE COMPOSITE EVENT command 107
ORGABCODE option
ASSIGN command 52
ORGANIZATION option
EXTRACT CERTIFICATE command 178
ORGANIZATLEN option
EXTRACT CERTIFICATE command 178
ORGUNIT option
EXTRACT CERTIFICATE command 178
ORGUNITLEN option
EXTRACT CERTIFICATE command 178
OUTDESCR option
SPOOLOPEN OUTPUT command 553
OUTDESCRERR condition
SPOOLOPEN OUTPUT command 556
OUTLINE operand
DFHMDF 857
DFHMDI 868
DFHMSD 877
OUTLINE option
ASSIGN command 52
OUTPARTN option
SEND CONTROL command 502
SEND MAP command 508
SEND TEXT command 522
SEND TEXT NOEDIT command 528
output control, 2980 General Banking Terminal System 425
output to common buffer, 2980 425
OVERFLOW condition
SEND MAP command 510
OWNER option
EXTRACT CERTIFICATE command 178

P

PA1–PA3 option
HANDLE AID command 264
PAGE option
SPOOLWRITE command 559
PAGENUM option
ASSIGN command 52
PAGING option
SEND CONTROL command 502
SEND MAP command 508
SEND TEXT command 522
SEND TEXT MAPPED command 525
SEND TEXT NOEDIT command 529
partition definition macro (DFHPDI) 856, 888
partition set definition macro (DFHPSD) 856, 889
PARTN operand
DFHMDI 868
DFHMSD 877
PARTN option
RECEIVE PARTN command 437
PARTNER option
ALLOCATE(APPC) command 35
CONNECT PROCESS command 80
GDS ALLOCATE command 211
GDS CONNECT PROCESS command 214
PARTNERIDERR condition
ALLOCATE (APPC) command 36
CONNECT PROCESS command 82
PARTNFAIL condition
RECEIVE MAP command 433
PARTNPAGE option
ASSIGN command 52
PARTNS option
ASSIGN command 52
PARTNSET option
ASSIGN command 52
PASSBK option
RECEIVE (non-z/OS Communications Server) command 428
SEND (non-z/OS Communications Server) command 497
passbook control, 2980 424
passing a session 308
passing control
expecting return (LINK) 323
on receipt of an AID (HANDLE AID command) 264
without return (XCTL) 747
passing data to new tasks 561
PASSTICKET option
REQUEST PASSTICKET 444
PASSWORD option
CHANGE PASSWORD command 73
SIGNON command 535
VERIFY PASSWORD command 615
WEB CONVERSE command 642
WEB SEND command (Client) 704
password phrase
WEB CONVERSE command 642
PASSWORDLEN option
WEB CONVERSE command 643
WEB SEND command (Client) 704
PATH option
WEB CONVERSE command 643
WEB EXTRACT or EXTRACT WEB command 188, 654
WEB PARSE URL command 663
WEB SEND command (Client) 704
PATHLENGTH option
WEB CONVERSE command 643
WEB EXTRACT or EXTRACT WEB command 188, 654
WEB PARSE URL command 663
WEB SEND command (Client) 705
PCT option
DUMP TRANSACTION command 155
performance, application, monitoring 341
PF1–24 option
HANDLE AID command 264
PFXLENG option
WRITE JOURNALNAME command 724
PGMIDERR condition
HANDLE ABEND command 264
INVOKE APPLICATION command 281
LINK ACQPROCESS command 334
LINK ACTIVITY command 338
LINK command 330
LOAD command 340
RELEASE command 439
START BREXIT command 573
XCTL command 749
PHRASE option
CHANGE PHRASE command 71
SIGNON command 535
VERIFY PHRASE command 618
PHRASELEN option
CHANGE PHRASE command 71
SIGNON command 535
VERIFY PHRASE command 618

PICIN operand
DFHMDF 857
PICOUT operand
DFHMDF 857
pipeline logical units 415, 484
PIPLENGTH option
CONNECT PROCESS command 81
EXTRACT PROCESS command 180
GDS CONNECT PROCESS command 214
GDS EXTRACT PROCESS command 217
PIPLIST option
CONNECT PROCESS command 81
EXTRACT PROCESS command 180
GDS CONNECT PROCESS command 214
GDS EXTRACT PROCESS command 217
PL/I language
argument values 6
LENGTH option default 7
PROCEDURE statement 12
STAE option 25
translated code 12
PLATFORM option 52
INVOKE APPLICATION command 280
plus 32K COMMAREAs (channels)
ASSIGN command 47
CHANNEL option of RETURN command 459
CHANNEL option of XCTL command 747
DELETE CHANNEL command 129
DELETE CONTAINER (CHANNEL) command 132
GET CONTAINER (CHANNEL) command 234
MOVE CONTAINER (CHANNEL) command 345
PUT CONTAINER (CHANNEL) command 357
QUERY CHANNEL command 365
START CHANNEL command 576
POINT command 348
POINT option
MONITOR command 342
pointer-ref argument, CICS command format 3
pointer-value argument, CICS command format 3
POOL option
DEFINE COUNTER command 108
DEFINE DCOUNTER command 108
DELETE COUNTER command 133
DELETE DCOUNTER command 133
GET COUNTER command 239
GET DCOUNTER command 239
QUERY COUNTER command 365
QUERY DCOUNTER command 365
REWIND COUNTER command 463
REWIND DCOUNTER command 463
UPDATE COUNTER command 610
UPDATE DCOUNTER command 610
POP HANDLE command 349
PORTNUMBER option
EXTRACT TCPIP command 184
WEB EXTRACT or EXTRACT WEB command 188, 654
WEB OPEN command 660
WEB PARSE URL command 663
PORTNUMNU option
EXTRACT TCPIP command 184
POS operand 857
DFHMDF 857
POST command 349
posting timer-event control area 350
PPT option
DUMP TRANSACTION command 155
PREDICATE option
GETNEXT EVENT command 256
INQUIRE EVENT command 275
PREFIX option
WRITE JOURNALNAME command 724
PRINCONVID option
GDS ASSIGN command 213
PRINSYSID option
ASSIGN command 52
GDS ASSIGN command 213
print displayed information 845
PRINT option
ISSUE ABORT command 289
ISSUE END command 297
ISSUE SEND command 318
ISSUE WAIT command 322
SEND CONTROL command 502
SEND MAP command 509
SEND MAP MAPPINGDEV command 513
SEND TEXT command 522
SEND TEXT NOEDIT command 529
SPOOLOPEN OUTPUT command 554
PRINT value
DFHMDI 868
DFHMSD 877
printer control character list, DFHBMSCA 849
priority of task, changing 74
PRIORITY option
CHANGE TASK command 74
PRIVACY option
EXTRACT TCPIP command 182
PROCESS option
ACQUIRE command 28
ASSIGN command 53
BUILD ATTACH (LUTYPE6.1) command 62
BUILD ATTACH (MRO) command 65
DEFINE PROCESS command 113
DELETE CONTAINER (BTS) command 131
EXTRACT ATTACH (LUTYPE6.1) command 169
EXTRACT ATTACH (MRO) command 172
GET CONTAINER (BTS) command 233
GETNEXT PROCESS command 257
PROCESS option *(continued)*
INQUIRE ACTIVITYID command 270
INQUIRE CONTAINER command 273
INQUIRE PROCESS command 276
PUT CONTAINER (BTS) command 356
STARTBROWSE ACTIVITY command 588
STARTBROWSE CONTAINER command 589
PROCESSBUSY condition
ACQUIRE command 29
CANCEL (BTS) command 69
DELETE CONTAINER (BTS) command 131
GET CONTAINER (BTS) command 234
LINK ACQPROCESS command 335
PUT CONTAINER (BTS) command 357
RESET ACQPROCESS command 445
RUN command 478
PROCESSERR condition
ACQUIRE command 29
CANCEL (BTS) command 69
DEFINE PROCESS command 114
GETNEXT PROCESS command 257
INQUIRE CONTAINER command 273
INQUIRE PROCESS command 276
LINK ACQPROCESS command 335
RESET ACQPROCESS command 445
RESUME command 453
RUN command 479
STARTBROWSE ACTIVITY command 588
STARTBROWSE CONTAINER command 590
STARTBROWSE PROCESS command 593
SUSPEND (BTS) command 594
processing state, of an activity
ACTIVE 270
CANCELLING 270
COMPLETE 270
DORMANT 270
INITIAL 270
processing task, control delay of 117
PROCESSTYPE option
ACQUIRE command 28
ASSIGN command 53
DEFINE PROCESS command 113
INQUIRE ACTIVITYID command 271
INQUIRE CONTAINER command 273
INQUIRE PROCESS command 276
STARTBROWSE ACTIVITY command 588
STARTBROWSE CONTAINER command 589
STARTBROWSE PROCESS command 592
PROCLENGTH option
CONNECT PROCESS command 81

PROCLENGTH option *(continued)*
EXTRACT PROCESS command 180
GDS CONNECT PROCESS command 214
GDS EXTRACT PROCESS command 217
PROCNAME option
CONNECT PROCESS command 81
EXTRACT PROCESS command 180
GDS CONNECT PROCESS command 214
GDS EXTRACT PROCESS command 217
PROFILE option
ALLOCATE (APPC) command 35
ALLOCATE (LUTYPE6.1) command 37
ALLOCATE (MRO) command 39
program control
commands 21
deleting loaded program 438
LINK command options 326
linking to another program 323
load a program, table, or map 338
returning program control 459
transfer program control 747
PROGRAM option
ASSIGN command 53
DEFINE ACTIVITY command 105
DEFINE PROCESS command 113
DUMP TRANSACTION command 155
HANDLE ABEND command 263
INQUIRE ACTIVITYID command 271
ISSUE LOAD command 306
LINK command 327
LOAD command 339
RELEASE command 438
XCTL command 748
PROT value
DFHMDF 857
PROTECT option
START command 568
PS operand
DFHMDF 857
DFHMDI 868
DFHMSD 877
PS option
ASSIGN command 53
PSEUDOBIN option
CONVERSE (non-z/OS Communications Server) command 99
RECEIVE (non-z/OS Communications Server) command 428
SEND (non-z/OS Communications Server) command 497
psid value
DFHMDF 857
DFHMDI 868
DFHMSD 877
ptr-ref64
command format 3
ptr-value64
command format 3
PUNCH option
SPOOLOPEN OUTPUT command 554
PURGE MESSAGE command 353
PURGEABILITY option
WAIT EXTERNAL command 627
WAITCICS command 634
PUSH HANDLE command 354
PUT CONTAINER (BTS) command 354
PUT CONTAINER (CHANNEL) command 357
PUT64 CONTAINER command 361

Q

QBUSY condition
READQ TD command 408
QIDERR condition
DELETEQ TD command 137
DELETEQ TS command 139
QUERY SECURITY command 372
READQ TD command 408
READQ TS command 411
WRITEQ TD command 730
WRITEQ TS command 734
QNAME option
ASSIGN command 53
DELETEQ TS command 138
READQ TS command 410
WRITEQ TS command 733
QUERY CHANNEL command 365
QUERY COUNTER command 365
QUERY DCOUNTER command 365
QUERY SECURITY command 369
QUERYPARM option
WEB READ QUERYPARM command 669
WEB READNEXT QUERYPARM command 673
WEB STARTBROWSE QUERYPARM command 711
QUERYSTRING option
WEB EXTRACT or EXTRACT WEB command 189, 655
WEB PARSE URL command 664
WEB SEND command 643
WEB SEND command (Client) 705
QUERYSTRLEN option
WEB EXTRACT or EXTRACT WEB command 189, 655
WEB PARSE URL command 664
WEB SEND command 643
WEB SEND command (Client) 705
QUEUE option
BUILD ATTACH (LUTYPE6.1) command 63
BUILD ATTACH (MRO) command 65
DELETEQ TD command 137
DELETEQ TS command 138
EXTRACT ATTACH (LUTYPE6.1) command 170
EXTRACT ATTACH (MRO) command 173
READQ TD command 406
READQ TS command 410
RETRIEVE command 454
QUEUE option *(continued)*
START command 568
WRITEQ TD command 729
WRITEQ TS command 733
QZERO condition
READQ TD command 408

R

RBA option
DELETE command 120
READ command 377
READNEXT command 390
READPREV command 400
RESETBR command 449
STARTBR command 583
WRITE command 717
RDATT condition
CONVERSE (non-z/OS Communications Server) command 101
RECEIVE (non-z/OS Communications Server) command 430
RECEIVE MAP command 433
reactivate an ABEND exit 262
READ command 372
READ option
QUERY SECURITY command 369
reading records
batch data interchange 313
browsing, next 385
browsing, previous (VSAM) 395
file control 372
from temporary storage queue 409
from terminal or LU 841
from transient data queue 405
READNEXT command 384
READPREV command 395
READQ TD command 405
READQ TS command 409
REALM option
WEB EXTRACT or EXTRACT WEB command 189, 655
REALMLEN option
WEB EXTRACT or EXTRACT WEB command 189, 655
RECEIVE (2260) command 423
RECEIVE (2980) command 424
RECEIVE (3270 logical) command 415
RECEIVE (3600 pipeline) command 415
RECEIVE (3600-3601) command 416
RECEIVE (3600-3614) command 416
RECEIVE (3650) command 417
RECEIVE (3767) command 417
RECEIVE (3770) command 418
RECEIVE (3790 3270-display) command 426
RECEIVE (3790 full-function or inquiry) command 418
RECEIVE (APPC) command 413
RECEIVE (LUTYPE2/LUTYPE3) command 413
RECEIVE (LUTYPE4) command 414
RECEIVE (LUTYPE6.1) command 414
RECEIVE (MRO) command 422
RECEIVE (non-z/OS Communications Server) command 422

RECEIVE (z/OS Communications Server default) command 412
RECEIVE command
input operation without data 846
read from terminal or logical unit 841
RECEIVE MAP command 430
RECEIVE MAP MAPPINGDEV command 433
RECEIVE PARTN command 435
RECFM option
BUILD ATTACH (LUTYPE6.1) command 63
BUILD ATTACH (MRO) command 65
EXTRACT ATTACH (LUTYPE6.1) command 170
EXTRACT ATTACH (MRO) command 173
RECORD option
BIF DIGEST command 60
RECORDBUSY condition
DELETE command 120
READ command 383
READNEXT command 395
READPREV command 405
REWRITE command 470
WRITE command 721
RECORDLEN option
BIF DIGEST command 60
RECORDLENGTH option
SPOOLOPEN OUTPUT command 554
records
deleting VSAM 120
reading 313, 372
release exclusive control 606
requesting next number 306
updating 315, 465
writing new 715
writing new (adding) 290
REDUCE option
GET COUNTER command 239
GET DCOUNTER command 239
register contents in assembler language 13
relative byte address (RBA) 120
RELEASE command 438
RELEASE option
SEND PAGE command 515
relocatable expression 7
REMOVE SUBEVENT command 439
REPEATABLE option
READ command 377
READNEXT command 390
READPREV command 400
REPLY option
WRITE OPERATOR command 727
REPLYLENGTH option
WRITE OPERATOR command 727
REQID option
CANCEL command 67
DELAY command 117
ENDBR command 159
POST command 352
READNEXT command 390
READPREV command 401
REQID option *(continued)*
RESETBR command 449
ROUTE command 473
SEND CONTROL command 502
SEND MAP command 509
SEND TEXT command 523
SEND TEXT MAPPED command 526
SEND TEXT NOEDIT command 529
START command 568
STARTBR command 583
WAIT JOURNALNAME command 629
WRITE JOURNALNAME command 724
REQUEST ENCRYPTPTKT command 440
REQUEST PASSTICKET command 443
REQUESTTYPE option
WEB EXTRACT or EXTRACT WEB command 189, 655
RESCLASS option
QUERY SECURITY command 370
RESET ACQPROCESS command 444
RESET ACTIVITY command 446
RESET option
HANDLE ABEND command 264
reset start for browse 447
RESETBR command 447
RESID option
QUERY SECURITY command 370
RESIDLENGTH option
QUERY SECURITY command 370
RESOURCE option
BUILD ATTACH (LUTYPE6.1) command 63
BUILD ATTACH (MRO) command 66
DEQ command 141
ENQ command 165
ENTER TRACENUM command 167
EXTRACT ATTACH (LUTYPE6.1) command 170
EXTRACT ATTACH (MRO) command 173
resource scheduling 139
RESP
deactivating NOHANDLE 267
option 11
values in EIBRESP 755
RESP2
EXPIRED in messages to console operators 728
INVREQ in messages to console operators 728
INVREQ in SIGNOFF command (Security control) 532
INVREQ in SIGNON (Security control) 535, 539
INVREQ in WAIT EXTERNAL 627
INVREQ on WAITCICS 634
LENGERR in messages to console operators 728
NOTAUTH in SIGNON (Security control) 536
option 11
RESP2 *(continued)*
USERIDERR in SIGNON (Security control) 536
values in EIBRESP2 755
Response Codes
of EXEC CICS commands 774
RESSEC option
ASSIGN command 53
RESTART option
ASSIGN command 53
RESTYPE option
QUERY SECURITY command 370
RESULT option
BIF DIGEST command 60
built-in function 60
RESUME command 451
RESUNAVAIL condition
LINK command 331
START command 561
START TRANSID (CHANNEL) command 576
RETAIN option
SEND PAGE command 516
RETCODE option
GDS ALLOCATE command 211
GDS ASSIGN command 213
GDS CONNECT PROCESS command 214
GDS EXTRACT ATTRIBUTES command 216
GDS EXTRACT PROCESS command 217
GDS FREE command 219
GDS ISSUE ABEND command 220
GDS ISSUE CONFIRMATION command 221
GDS ISSUE ERROR command 223
GDS ISSUE PREPARE command 224
GDS ISSUE SIGNAL command 225
GDS RECEIVE command 227
GDS SEND command 229
GDS WAIT command 231
RETPAGE condition
SEND CONTROL command 503
SEND MAP command 511
SEND PAGE command 517
SEND TEXT command 524
RETRIEVE command 453
retrieve data stored for task 453
RETRIEVE REATTACH EVENT command 456
RETRIEVE SUBEVENT command 458
RETURN command 459
return program control 459
RETURNPROG option
ASSIGN command 53
REVERSE value
DFHMDF 857
DFHMDI 868
DFHMSD 877
REWIND COUNTER command 463
REWIND DCOUNTER command 463
REWRITE command 465
REWRITE option
WRITEQ TS command 733
RIDFLD option
DELETE command 120

RIDFLD option *(continued)*
ISSUE ADD command 291
ISSUE ERASE command 302
ISSUE NOTE command 307
ISSUE REPLACE command 316
READ command 377
READNEXT command 390
READPREV command 401
RESETBR command 449
STARTBR command 583
WRITE command 717
RIGHT value
DFHMDF 857
DFHMDI 868
ROLLBACK option
SYNCPOINT ROLLBACK command 596
ROLLEDBACK condition
LINK command 331
SYNCPOINT command 595
ROUTE command 471
ROUTECODES option
WRITE OPERATOR command 727
RPROCESS option
BUILD ATTACH (LUTYPE6.1) command 63
BUILD ATTACH (MRO) command 66
EXTRACT ATTACH (LUTYPE6.1) command 170
EXTRACT ATTACH (MRO) command 173
RRESOURCE option
BUILD ATTACH (LUTYPE6.1) command 63
BUILD ATTACH (MRO) command 66
EXTRACT ATTACH (LUTYPE6.1) command 170
EXTRACT ATTACH (MRO) command 173
RRN option
DELETE command 120
ISSUE ADD command 291
ISSUE ERASE command 302
ISSUE NOTE command 307
ISSUE REPLACE command 316
READ command 378
READNEXT command 391
READPREV command 401
RESETBR command 449
STARTBR command 584
WRITE command 718
RTEFAIL condition
ROUTE command 474
RTERMID option
RETRIEVE command 454
START command 568
RTESOME condition
ROUTE command 474
RTRANSID option
RETRIEVE command 454
START command 568
RUN command 475

S

SAA (Systems Application Architecture) communications (CPI) 847
Resource Recovery 847
SADDRLENGTH option
EXTRACT TCPIP command 184
SAME value
DFHMDI 868
schedule use of resource by task 139, 163
scheduling commands 21
SCHEME option
WEB EXTRACT or EXTRACT WEB command 189, 655
WEB OPEN command 660
SCHEMENAME option
WEB PARSE URL command 664
SCRNHT option
ASSIGN command 54
SCRNWD option
ASSIGN command 54
SCS (SNA character string)
CONVERSE command 85
SEND (z/OS Communications Server) command 489
SEND command 483
SCS printer logical unit, 3790 489
SECONDS option
DEFINE TIMER command 116
DELAY command 117
POST command 352
ROUTE command 473
START command 569
security commands 21
SEGMENTLIST option
DUMP TRANSACTION command 155
SELNERR condition
ISSUE ABORT command 289
ISSUE ADD command 291
ISSUE END command 298
ISSUE ERASE command 302
ISSUE NOTE command 307
ISSUE QUERY command 313
ISSUE REPLACE command 317
ISSUE SEND command 319
ISSUE WAIT command 323
SEND (2260) command 495
SEND (2980) command 495
SEND (3270 logical) command 484
SEND (3600 pipeline) command 484
SEND (3600-3601) command 485
SEND (3600-3614) command 486
SEND (3650 interpreter) command 486
SEND (3650-3270) command 487
SEND (3650-3653) command 487
SEND (3650-3680) command 488
SEND (3767) command 488
SEND (3770) command 488
SEND (3790 3270-display) command 490
SEND (3790 3270-printer) command 490
SEND (3790 full-function or inquiry) command 489
SEND (3790 SCS) command 489
SEND (APPC) command 481
SEND (LUTYPE2/LUTYPE3) command 482
SEND (LUTYPE4) command 482
SEND (LUTYPE6.1) command 483
SEND (MRO) command 495
SEND (non-z/OS Communications Server default) command 494
SEND (SCS) command 483
SEND (z/OS Communications Server default) command 481
send asynchronous interrupt 841
SEND command
write to terminal 841
SEND CONTROL command 499
SEND MAP command 504
SEND MAP MAPPINGDEV command 511
SEND PAGE command 514
SEND PARTNSET command 517
SEND TEXT command 518
SEND TEXT MAPPED command 525
SEND TEXT NOEDIT command 527
sending data to output device 317
sequential retrieval, browsing
reading records 372
SERIALNUM option
EXTRACT CERTIFICATE command 178
SERIALNUMLEN option
EXTRACT CERTIFICATE command 179
SERVADDRNU option
EXTRACT TCPIP command 184
SERVERADDR option
EXTRACT TCPIP command 184
SERVERCONV option
WEB RECEIVE command (Server) 679
WEB SEND command (Server) 694
SERVERNAME option
EXTRACT TCPIP command 184
SESSBUSY condition
ALLOCATE (LUTYPE6.1) command 38
SESSION option
ALLOCATE (LUTYPE6.1) command 37
CONNECT PROCESS command 81
CONVERSE (non-z/OS Communications Server) command 99
CONVERSE (z/OS Communications Server) command 94
EXTRACT ATTACH (LUTYPE6.1) command 171
EXTRACT ATTACH (MRO) command 174
EXTRACT ATTRIBUTES (MRO) command 176
FREE (LUTYPE6.1) command 203
FREE (MRO) command 203
ISSUE DISCONNECT (LUTYPE6.1) command 296
ISSUE SIGNAL (LUTYPE6.1) command 321
POINT command 348
RECEIVE (non-z/OS Communications Server) command 428

SESSION option *(continued)*
RECEIVE (z/OS Communications Server) command 420
SEND (non-z/OS Communications Server) command 497
SEND (z/OS Communications Server) command 492
WAIT TERMINAL command 632
session, passing 308
SESSIONERR condition
ALLOCATE (LUTYPE6.1) command 38
EIBRCODE bytes 1-2 755
SESSTOKEN option
WEB CLOSE command 637
WEB CONVERSE command 643
WEB ENDBROWSE HTTPHEADER command 651
WEB EXTRACT or EXTRACT WEB command 189, 655
WEB OPEN command 660
WEB READ HTTPHEADER command 667
WEB READNEXT HTTPHEADER command 672
WEB RECEIVE command (Client) 685
WEB SEND command (Client) 705
WEB STARTBROWSE HTTPHEADER command 710
WEB WRITE HTTPHEADER command 714
SET option
ADDRESS SET command 33
CONVERSE (non-z/OS Communications Server) command 100
CONVERSE (z/OS Communications Server) command 94
EXTRACT LOGONMSG command 180
GDS RECEIVE command 227
GET CONTAINER (BTS) command 233
GET CONTAINER (CHANNEL) command 237
GET64 CONTAINER command 261
GETMAIN command 247
GETMAIN64 command 251
INQUIRE CONTAINER command 273
ISSUE RECEIVE command 314
LOAD command 339
POST command 352
READ command 378
READNEXT command 391
READPREV command 401
READQ TD command 407
READQ TS command 410
RECEIVE (non-z/OS Communications Server) command 428
RECEIVE (z/OS Communications Server) command 420
RECEIVE MAP command 432
RECEIVE MAP MAPPINGDEV command 435
RECEIVE PARTN command 437
SET option *(continued)*
RETRIEVE command 454
SEND CONTROL command 502
SEND MAP command 509
SEND MAP MAPPINGDEV command 513
SEND PAGE command 516
SEND TEXT command 523
WEB CONVERSE command 645
WEB READ FORMFIELD command 666
WEB READ QUERYPARM command 669
WEB RECEIVE command (Client) 685
WEB RECEIVE command (Server) 679
SHARED option
GETMAIN command 247
GETMAIN64 command 251
SIGDATA option
ASSIGN command 54
SIGNAL condition
CONVERSE (z/OS Communications Server) command 96
ISSUE CONFIRMATION command 293
ISSUE DISCONNECT (default) command 295
ISSUE ERROR command 305
RECEIVE (z/OS Communications Server) command 422
SEND (z/OS Communications Server) command 494
WAIT SIGNAL command 631
WAIT TERMINAL command 632
SIGNAL EVENT command 530
SIGNOFF command 532
SIGNON command 533
SIGNON TOKEN command 537
single thread used with JES 550
SIT option
DUMP TRANSACTION command 155
SIZE operand
DFHMDI 868
SNAMELENGTH option
EXTRACT TCPIP command 184
SOAPFAULT ADD command 541
SOAPFAULT CREATE command 543
SOAPFAULT DELETE command 546
SOSI operand
DFHMDF 857
DFHMDI 868
DFHMSD 877
SOSI option
ASSIGN command 54
SPCOMMAND
RESID value not valid 372
SPOLBUSY condition
SPOOLOPEN INPUT command 551
SPOOLOPEN OUTPUT command 556
SPOLERR condition
SPOOLOPEN INPUT command 551
SPOOLREAD command 559
SPOOLWRITE command 561
Spool commands 22
SPOOLCLOSE command 547
SPOOLCLOSE, implicit 550
SPOOLOPEN INPUT command 549
SPOOLOPEN OUTPUT 552
SPOOLREAD command 556
SPOOLWRITE command 559
SRVRADDR6NU option
EXTRACT TCPIP command 184
SRVRIPFAMILY option
EXTRACT TCPIP command 184
SSLTYPE option
EXTRACT TCPIP command 185
STAE option, PL/I 25
standard attribute and printer control character list, BMS (DFHBMSCA) 849
START ATTACH command 572
START CHANNEL command 576
START command 561, 573
STARTBR command 580
STARTBROWSE ACTIVITY command 587
STARTBROWSE CONTAINER (CHANNEL) command 590
STARTBROWSE CONTAINER command 588
STARTBROWSE EVENT command 591
STARTBROWSE PROCESS command 592
STARTCODE option
ASSIGN command 54
STATE option
ALLOCATE (APPC) command 35
ALLOCATE (MRO) command 39
CONNECT PROCESS command 81
CONVERSE (non-z/OS Communications Server) command 100
CONVERSE (z/OS Communications Server) command 94
EXTRACT ATTRIBUTES (APPC) command 174
EXTRACT ATTRIBUTES (MRO) command 176
EXTRACT CERTIFICATE command 179
FREE (APPC) command 201
FREE (MRO) command 204
GDS ALLOCATE command 211
GDS CONNECT PROCESS command 214
GDS EXTRACT ATTRIBUTES command 216
GDS FREE command 219
GDS ISSUE ABEND command 220
GDS ISSUE CONFIRMATION command 221
GDS ISSUE ERROR command 223
GDS ISSUE PREPARE command 224
GDS ISSUE SIGNAL command 225
GDS RECEIVE command 227
GDS SEND command 229
GDS WAIT command 231
ISSUE ABEND command 287
ISSUE CONFIRMATION command 292
ISSUE ERROR command 304

STATE option *(continued)*
ISSUE PREPARE command 310
ISSUE SIGNAL (APPC) command 320
RECEIVE (non-z/OS Communications Server) command 429
RECEIVE (z/OS Communications Server) command 420
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 492
WAIT CONVID command 623
STATELEN option
EXTRACT CERTIFICATE command 179
STATIONID option
ASSIGN command 55
STATUS option
CHECK TIMER command 79
INQUIRE TIMER command 277
STATUSCODE option
WEB CONVERSE command 645
WEB RECEIVE command (Client) 685
WEB SEND command (Server) 695
STATUSLEN option
WEB CONVERSE command 645
WEB RECEIVE command (Client) 686
WEB SEND command (Server) 696
STATUSTEXT option
WEB CONVERSE command 645
WEB RECEIVE command (Client) 686
WEB SEND command (Server) 696
storage area length 41
storage control commands 22
STORAGE operand
DFHMSD 877
STORAGE option
DUMP TRANSACTION command 155
STRELERR condition
SPOOLCLOSE command 549
SPOOLOPEN INPUT command 552
SPOOLOPEN OUTPUT command 556
SPOOLREAD command 559
SPOOLWRITE command 561
STRFIELD option
CONVERSE (non-z/OS Communications Server) command 100
CONVERSE (z/OS Communications Server) command 94
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 493
STRINGFORMAT option
FORMATTIME command 198
STRINGZONE option
FORMATTIME command 199
stub, program 13
SUBADDR option
ISSUE ABORT command 289
SUBADDR option *(continued)*
ISSUE END command 297
ISSUE SEND command 318
ISSUE WAIT command 322
SUBEVENT option
ADD SUBEVENT command 30
DEFINE COMPOSITE EVENT command 107
REMOVE SUBEVENT command 440
RETRIEVE SUBEVENT command 459
SUFFIX operand
DFHMSD 877
DFHPSD 889
SUPPRESSED condition
DUMP TRANSACTION command 158
GET COUNTER command 239
REWIND COUNTER command 463
UPDATE COUNTER command 610
WRITE command 721
SUSPEND (BTS) command 593
SUSPEND command 593
SUSPSTATUS option
CHECK ACQPROCESS command 76
CHECK ACTIVITY command 78
INQUIRE ACTIVITYID command 271
switched line disconnection 841
SYMBOL option
DOCUMENT INSERT command 147
DOCUMENT SET command 152
SYMBOLERR condition
DOCUMENT SET command 153
SYMBOLLIST option
DOCUMENT SET command 143, 152
synchronization levels
basic conversations 218
synchronize, action
journal output (WAIT JOURNALNAME) 628
terminal input/output 841
SYNCHRONOUS option
RUN command 477
SYNCLEVEL option
CONNECT PROCESS command 81
EXTRACT PROCESS command 180
GDS CONNECT PROCESS command 214
GDS EXTRACT PROCESS command 218
SYNCONRETURN option
LINK command 327
syncpoint
backing out 596
commands 22
establishing 595
SYNCPOINT command 595
SYNCPOINT ROLLBACK command 596
syntax notation 2
SYSBUSY condition
ALLOCATE (APPC) command 36
ALLOCATE (LUTYPE6.1) command 38
ALLOCATE (MRO) command 39
EIBRCODE byte 3 755
SYSID option
ALLOCATE (APPC) command 35
ALLOCATE (LUTYPE6.1) command 37
ALLOCATE (MRO) command 39
ASSIGN command 55
CANCEL command 67
DELETE command 120
DELETEQ TD command 137
DELETEQ TS command 138, 733
ENDBR command 159
EXTRACT TCT command 186
GDS ALLOCATE command 211
LINK command 328
READ command 378
READNEXT command 391
READPREV command 401
READQ TD command 407
READQ TS command 410
RESETBR command 449
REWRITE command 467
START command 569
START TRANSID (CHANNEL) command 576
STARTBR command 584
UNLOCK command 608
WRITE command 718
WRITEQ TD command 729
SYSIDERR condition
ALLOCATE (APPC) command 36
ALLOCATE (LUTYPE6.1) command 38
ALLOCATE (MRO) command 40
CANCEL command 68
DELETE command 120
DELETEQ TD command 137
DELETEQ TS command 139
EIBRCODE bytes 1-2 755
ENDBR command 160
LINK command 331
READ command 383
READNEXT command 395
READPREV command 405
READQ TD command 408
READQ TS command 411
RESETBR command 451
REWRITE command 470
START command 561
START TRANSID (CHANNEL) command 576
STARTBR command 586
UNLOCK command 610
WRITE command 721
WRITEQ TD command 730
WRITEQ TS command 734
systemname
definition 4, 5, 6, 8, 9
systemname argument, CICS command format 3

T

TABLES option
DUMP TRANSACTION command 156
tables, loading 338

task
initiation 561
task control commands 22
TASK option
DUMP TRANSACTION command 156
task, abnormal termination 262
task, delay processing of 117
TASKDATALOC resource definition option 32
TASKPRIORITY option
ASSIGN command 55
TCP/IP services 22
TCPIPSERVICE option
EXTRACT TCPIP command 185
TCT option
DUMP TRANSACTION command 156
TCTUA option
ADDRESS command 32
TCTUALENG option
ASSIGN command 55
teletypewriter
messages 843
programming 843
TELLERID option
ASSIGN command 55
TEMPLATE option
DOCUMENT INSERT command 148
temporary storage control commands 22
TERM operand
DFHMDI 868
DFHMSD 877
TERMCODE option
ASSIGN command 55, 57
TERMERR condition
CONNECT PROCESS command 82
CONVERSE (non-z/OS Communications Server) command 101
CONVERSE (z/OS Communications Server) command 96
ISSUE ABEND command 288
ISSUE CONFIRMATION command 293
ISSUE COPY (3270 logical) command 294
ISSUE DISCONNECT (default) command 295
ISSUE DISCONNECT (LUTYPE6.1) command 296
ISSUE EODS command 300
ISSUE ERASEAUP command 303
ISSUE ERROR command 305
ISSUE LOAD command 306
ISSUE PREPARE command 311
ISSUE PRINT command 312
ISSUE SIGNAL (APPC) command 320
ISSUE SIGNAL (LUTYPE6.1) command 321
LINK command 332
RECEIVE (non-z/OS Communications Server) command 430
RECEIVE (z/OS Communications Server) command 422
TERMERR condition *(continued)*
SEND (non-z/OS Communications Server) command 499
SEND (z/OS Communications Server) command 494
WAIT SIGNAL command 631
WAIT TERMINAL command 633
TERMID option
EXTRACT TCT command 186
ISSUE COPY (3270 logical) command 294
START command 569
START TRANSID (CHANNEL) command 576
TERMIDERR condition
START command 561
START TRANSID (CHANNEL) command 576
terminal control 841
commands 22
terminal model codes 57
terminal operator paging, initiate paging transaction 514
TERMINAL option
DUMP TRANSACTION command 156
RECEIVE MAP command 432
SEND CONTROL command 503
SEND MAP command 509
SEND TEXT command 523
SEND TEXT MAPPED command 526
SEND TEXT NOEDIT command 529
terminal type codes 57
terminate data set processing
abnormal 288
normal 297
TERMPRIORITY option
ASSIGN command 55
TERMTHDACT 25
TEST EVENT command 597
TEXT option
DOCUMENT INSERT command 148
WRITE OPERATOR command 727
TEXTKYBD option
ASSIGN command 55
TEXTLENGTH option
WRITE OPERATOR command 728
TEXTPRINT option
ASSIGN command 55
threadsafe commands 751
time of day, requesting 40
TIME option
DELAY command 117
FORMATTIME command 199
POST command 352
ROUTE command 473
START command 561
TIMEDOUT condition
WEB CONVERSE command 650
WEB EXTRACT or EXTRACT WEB command (Client) 191, 657
WEB OPEN command 662
WEB RECEIVE command (Client) 688
TIMEOUT option
FETCH CHILD command 192, 193
TIMEOUT option *(continued)*
WRITE OPERATOR command 728
TIMER option
CHECK TIMER command 80
DEFINE TIMER command 116
DELETE TIMER command 136
FORCE TIMER command 195
GETNEXT EVENT command 256
INQUIRE EVENT command 275
INQUIRE TIMER command 277
timer-event control area 350
TIMERERR condition
CHECK TIMER command 80
DEFINE TIMER command 117
DELETE TIMER command 136
FORCE TIMER command 195
INQUIRE TIMER command 278
TIMESEP option
FORMATTIME command 199
TIOAPFX operand
DFHMDI 868
DFHMSD 877
TITLE option
ROUTE command 474
TO option
DOCUMENT INSERT command 148
TOACTIVITY option
MOVE CONTAINER (BTS) command 345
TOCHANNEL option
MOVE CONTAINER (CHANNEL) command 347
WEB CONVERSE command 645
WEB RECEIVE command (Client) 686
WEB RECEIVE command (Server) 680
TOCONTAINER option
WEB CONVERSE command 646
WEB RECEIVE command (Client) 686
WEB RECEIVE command (Server) 680
TOFLENGTH option
CONVERSE (non-z/OS Communications Server) command 100
CONVERSE (z/OS Communications Server) command 94
fullword alternative to TOLENGTH 841
SPOOLREAD command 557
TOKEN option
DELETE command 120
READ command 378
READNEXT 391
READPREV command 401
REWRITE command 467
SIGNON TOKEN command 539
SPOOLCLOSE command 548
SPOOLOPEN INPUT command 550
SPOOLOPEN OUTPUT command 554
SPOOLREAD command 557
SPOOLWRITE command 560
UNLOCK command 608
VERIFY TOKEN command 622

TOKENERR condition
ENDBROWSE ACTIVITY command 161
ENDBROWSE CONTAINER (CHANNEL) command 162
ENDBROWSE CONTAINER command 161
ENDBROWSE EVENT command 162
ENDBROWSE PROCESS command 163
GETNEXT ACTIVITY command 253
GETNEXT CONTAINER (CHANNEL) command 255
GETNEXT CONTAINER command 254
GETNEXT EVENT command 256
GETNEXT PROCESS command 257
WEB CONVERSE command 650
WEB SEND command (Client) 708
TOKENLEN option
SIGNON TOKEN command 539
VERIFY TOKEN command 622
TOKENTYPE option
SIGNON TOKEN command 539
VERIFY TOKEN command 622
TOLENGTH option
CONVERSE (non-z/OS Communications Server) command 100
CONVERSE (z/OS Communications Server) command 94
fullword length alternative (TOFLENGTH) 841
WEB CONVERSE command 646
TOPROCESS option
MOVE CONTAINER (BTS) command 345
TRACENUM option
ENTER TRACENUM command 167
TRAILER operand
DFHMDI 868
TRAILER option
SEND PAGE command 516
SEND TEXT command 523
TRANPRIORITY option
ASSIGN command 55
transfer program control 747
TRANSFORM DATATOXML command 599
TRANSFORM XMLTODATA command 603
TRANSID option
CANCEL command 67
DEFINE ACTIVITY command 105
DEFINE PROCESS command 113
INQUIRE ACTIVITYID command 271
LINK command 328
RETURN command 461
RUN TRANSID command 480
SEND PAGE command 516
START ATTACH command 572
START BREXITcommand 573
START command 561
START TRANSID (CHANNEL) command 576
TRANSIDERR condition
DEFINE ACTIVITY command 106
DEFINE PROCESS command 114
START ATTACH command 572
START BREXIT command 573
START command 561
START TRANSID (CHANNEL) command 576
transient data commands 24
transient data control
delete intrapartition queue 136
read data from TD queue 405
write data to TD queue 729
translated code 12
TRANSP operand
DFHMDF 857
DFHMDI 868
DFHMSD 877
TRIGGER option
HANDLE AID command 264
TRIGGER value
DFHMDF 857
DFHMDI 868
DFHMSD 877
TRIGRAPH operand
DFHMSD 877
TRT option
DUMP TRANSACTION command 157
TSIOERR condition
PURGE MESSAGE command 353
SEND CONTROL command 503
SEND MAP command 511
SEND PAGE command 517
SEND TEXT command 524
SEND TEXT MAPPED command 526
SEND TEXT NOEDIT command 530
TWA option
ADDRESS command 32
TWALENG option
ASSIGN command 55
type codes (terminal) 57
TYPE operand
DFHMSD 877
TYPE option
WEB RECEIVE command (Server) 680
TYPENAME option
TRANSFORM XMLTODATA command 604
TYPENAMELEN option
TRANSFORM XMLTODATA command 605
TYPENS option
TRANSFORM XMLTODATA command 605
TYPENSLEN option
TRANSFORM XMLTODATA command 605

U

UNATTEND option
ASSIGN command 55
UNCOMMITTED
READ command 378
UNCOMMITTED option
READNEXT 391
READPREV command 402
UNDERLINE value
DFHMDF 857
DFHMDI 868
DFHMSD 877
UNEXPIN condition
ISSUE ABORT command 290
ISSUE ADD command 291
ISSUE END command 298
ISSUE ERASE command 302
ISSUE NOTE command 307
ISSUE QUERY command 313
ISSUE RECEIVE command 315
ISSUE REPLACE command 317
ISSUE SEND command 319
ISSUE WAIT command 323
RECEIVE MAP command 433
UNLOCK command 606
UNPROT value
DFHMDF 857
UNTIL option
DELAY command 117
UPDATE COUNTER command 610
UPDATE DCOUNTER command 610
UPDATE option
QUERY SECURITY command 371
READ command 379
READNEXT 392
READPREV command 402
updating records
batch data interchange 315
file control 465
URIMAP option
WEB EXTRACT or EXTRACT WEB command 190, 656
WEB OPEN command 643, 661, 705
URL option
WEB PARSE URL command 664
URLLENGTH option
WEB PARSE URL command 664
USERDATAKEY option
GETMAIN command 247
GETMAIN64 command 251
USEREXIT value
DFHMDF 857
DFHMDI 868
DFHMSD 877
USERID option
ASSIGN command 55
CHANGE PASSWORD command 73
CHANGE PHRASE command 71
DEFINE ACTIVITY command 105
DEFINE PROCESS command 113
EXTRACT CERTIFICATE command 179
INQUIRE ACTIVITYID command 271
SIGNON command 535
SPOOLOPEN INPUT command 550
SPOOLOPEN OUTPUT command 554
START BREXITcommand 573
START command 561
START TRANSID (CHANNEL) command 576

USERID option *(continued)*
VERIFY PASSWORD command 615
VERIFY PHRASE command 618
USERIDERR condition
CHANGE PASSWORD command 72, 73
SIGNON command 536
START BREXIT command 573
START command 561
START TRANSID (CHANNEL) command 576
VERIFY PASSWORD command 616
VERIFY PHRASE command 619
USERNAME option
ASSIGN command 56
WEB CONVERSE command 644
WEB SEND command (Client) 705
USERNAMELEN option
WEB CONVERSE command 644
WEB SEND command (Client) 706
USERPRIORITY option
ASSIGN command 56
USING option
ADDRESS SET command 33
UTC 199

V

VALIDATION option
ASSIGN command 56
VALIDN operand
DFHMDF 857
DFHMDI 868
DFHMSD 877
VALUE option
DEFINE COUNTER command 108
DEFINE DCOUNTER command 108
DOCUMENT SET command 152
GET COUNTER command 239
GET DCOUNTER command 239
QUERY COUNTER command 365
QUERY DCOUNTER command 365
UPDATE COUNTER command 610
UPDATE DCOUNTER command 610
WEB READ FORMFIELD command 666
WEB READ HTTPHEADER command 668
WEB READ QUERYPARM command 670
WEB READNEXT FORMFIELD command 671
WEB READNEXT HTTPHEADER command 672
WEB READNEXT QUERYPARM command 674
WEB WRITE HTTPHEADER command 714
VALUELENGTH option
WEB READ FORMFIELD command 666
WEB READ HTTPHEADER command 668
WEB READ QUERYPARM command 670
WEB READNEXT FORMFIELD command 671
VALUELENGTH option *(continued)*
WEB READNEXT HTTPHEADER command 672
WEB READNEXT QUERYPARM command 674
WEB WRITE HTTPHEADER command 714
VERIFY PASSWORD command 613
VERIFY PHRASE command 616
VERIFY TOKEN command 619
VERSIONLEN option
WEB EXTRACT or EXTRACT WEB command 190, 656
VIEWPOS operand
DFHPDI 888
VIEWSZE operand
DFHPDI 888
VOLUME option
ISSUE ABORT command 289
ISSUE ADD command 291
ISSUE END command 298
ISSUE ERASE command 302
ISSUE NOTE command 307
ISSUE QUERY command 312
ISSUE REPLACE command 316
ISSUE SEND command 318
ISSUE WAIT command 323
VOLUMELENG option
ISSUE ABORT command 289
ISSUE ADD command 291
ISSUE END command 298
ISSUE ERASE command 302
ISSUE NOTE command 307
ISSUE QUERY command 312
ISSUE REPLACE command 316
ISSUE SEND command 318
ISSUE WAIT command 323
VSAM WRITE MASSINSERT
DISABLED cannot occur 608
NOTOPEN cannot occur 609
terminate operation 607
VTAB operand
DFHMSD 877

W

WAIT CONVID (APPC) command 623
WAIT EVENT command 624
WAIT EXTERNAL command 626
WAIT JOURNALNAME command 628
WAIT JOURNALNUM command 630
WAIT option
GDS SEND command 230
ISSUE COPY (3270 logical) command 294
ISSUE ERASEAUP command 303
RETRIEVE command 454
SEND (non-z/OS Communications Server) command 498
SEND (z/OS Communications Server) command 493
SEND command 841
SEND CONTROL command 503
SEND MAP command 509
SEND TEXT command 523
SEND TEXT MAPPED command 526
WAIT option *(continued)*
SEND TEXT NOEDIT command 529
terminal control 841
WRITE JOURNALNAME command 724
WAIT SIGNAL command 631
WAIT TERMINAL command 632
general information 841
WAITCICS command 633
waits
batch data interchange 322
for event to occur 624
terminal control operation 841
WEB CLOSE command 635
WEB CONVERSE command 637
WEB ENDBROWSE FORMFIELD command 650
WEB ENDBROWSE HTTPHEADER command 651
WEB ENDBROWSE QUERYPARM command 651
WEB EXTRACT command 652
WEB OPEN command 657
WEB PARSE URL command 662
WEB READ FORMFIELD command 664
WEB READ HTTPHEADER command 667
WEB READ QUERYPARM command 668
WEB READNEXT FORMFIELD command 670
WEB READNEXT HTTPHEADER command 671
WEB READNEXT QUERYPARM command 673
WEB RECEIVE command (Client) 681
WEB RECEIVE command (Server) 674
WEB RETRIEVE command 688
WEB SEND command (Client) 698
WEB SEND command (Server) 689
web services commands 25
WEB STARTBROWSE FORMFIELD command 708
WEB STARTBROWSE HTTPHEADER command 710
WEB STARTBROWSE QUERYPARM command 711
web support 25
WEB WRITE HTTPHEADER command 712
WPMEDIA option
ISSUE ABORT command 289
ISSUE END command 298
ISSUE SEND command 318
ISSUE WAIT command 323
WRAP option
GET COUNTER command 239
GET DCOUNTER command 239
WRBRK condition
CONVERSE (non-z/OS Communications Server) command 102
SEND (non-z/OS Communications Server) command 499
SEND CONTROL command 504
SEND MAP command 511
SEND PAGE command 517

WRBRK condition *(continued)*
SEND TEXT command 524
SEND TEXT MAPPED command 526
SEND TEXT NOEDIT command 530
WRITE command 715
WRITE JOURNALNAME command 722
WRITE JOURNALNUM command 726
WRITE OPERATOR command 726
critical action 727
eventual action 727
immediate action 727
WRITEQ TD command 729
WRITEQ TS command 731
writing data
to temporary storage queue 731
to terminal or logical unit 841
to transient data queue 729
writing records to data sets
batch data interchange 290
file control 715
WSACONTEXT BUILD command 735
WSACONTEXT DELETE command 739
WSACONTEXT GET command 740
WSAEPR CREATE command 744

X

XCTL command 747
XINIT operand
DFHMDF 857
XMLCONTAINER option
TRANSFORM XMLTODATA command 605
XMLTRANSFORM option
TRANSFORM XMLTODATA command 605
XRBA option
DELETE command 120
READ command 379
READNEXT command 392
READPREV command 402
RESETBR command 449
STARTBR command 584
WRITE command 718
XRF, generic applid 44

Y

YEAR option
DEFINE TIMER command 116
FORMATTIME command 199
YES value
DFHMDI 868
DFHMSD 877
YYDDD option
FORMATTIME command 199
YYDDMM option
FORMATTIME command 199
YYMMDD option
FORMATTIME command 199
YYYYDDD option
FORMATTIME command 199
YYYYDDMM option
FORMATTIME command 200
YYYYMMDD option
FORMATTIME command 200

Z

z/OS Communications Server 83
z/OS Communications Server logon data, access to 179
ZERO value
DFHMDF 857

