This excerpt from Murach's CICS Desk Reference gives you an idea of the material that’s included in Section 2 of the book (Units 7 and 8), entitled “CICS Command Reference.” It begins with all of the summary information found in Unit 7. Then, it contains the introduction to Unit 8, along with the reference information for 8 of the 121 commands that are presented in that unit: DEFINE COUNTER, DOCUMENT CREATE, ENQ, LINK, READNEXT, SEND MAP, START, and WEB READ FORMFIELD.
CICS command preview

This unit presents information that will help you use the CICS command reference material in Unit 8. First, it gives you a summary of the syntax conventions that are used for all the CICS commands. Second, it presents two error-handling options—RESP and RESP2—that can be coded on any CICS command. And finally, it lists the commands in Unit 8 by function. So if you're not sure which command you're looking for, you can check here before going to the alphabetical list in Unit 8.

Syntax conventions .............................................................. 186
Syntax notation ................................................................................................ 186
Argument values .............................................................................................. 186
Data types ........................................................................................................ 187
RESP and RESP2 options .................................................... 188
Commands by function ........................................................ 189
Syntax conventions

Syntax notation

Unit 8 presents the syntax for each of the CICS commands as well as an explanation of what each command does. In order to accurately represent each command, the following syntax methods are used:

UPPERCASE Indicates CICS code that must be entered as shown.

option Indicates an option that must be coded.

[ option ] Indicates an option that may be coded but that is not required in order for the command to successfully execute.

{ option | option } Indicates a set of alternative options, one of which must be coded.

[ option | option ] Indicates a set of alternative options, one of which may be coded.

{ option | option } Indicates a set of alternative options, one of which must be coded.

option Indicates the default option.

option-1 Indicates that option-2 is coded only in conjunction with option-1.

... Indicates that the preceding option may be repeated multiple times.

Argument values

Most CICS commands require you to code one or more options. And most of the options require you to supply a value, or argument, in parentheses. The command syntax specifies which of these types of values you can code for each option:

data-value A COBOL data name coded in the Working-Storage or Linkage Section or a literal. The option may require a binary halfword, a binary fullword, an unsigned binary doubleword, or a character string.

data-area A COBOL data name coded in the Working-Storage or Linkage Section. The option may require a binary halfword, a binary fullword, an unsigned binary doubleword, or a character string.

cvda A CICS-value data area defined in the Working-Storage or Linkage Section as a binary fullword. The value typically describes the state of a particular resource. You can use the DFHVALUE keyword to work with cvda values.
pointer-ref The name of a BLL cell coded in the Linkage Section. A pointer-ref must be a binary fullword. Use the COBOL ADDRESS OF special register to set the reference.

pointer-value The name of a BLL cell coded in the Linkage Section or the name of a Working-Storage Section field that contains a BLL cell value. A pointer-value must be a binary fullword. Use the COBOL ADDRESS OF special register to set the value.

name An alphanumeric literal or the name of a Working-Storage or Linkage Section field that contains the value to be used.

filename An alphanumeric literal or a Working-Storage or Linkage Section field that specifies the name of a file.

systemname An alphanumeric literal or a Working-Storage or Linkage Section field that specifies a remote system ID.

label The name of a paragraph or section.

hhmmss A numeric literal or the name of a seven-digit packed-decimal field (PIC S9(7) COMP-3) defined in the Working-Storage or Linkage Section. $hh$ represents hours, $mm$ represents minutes, and $ss$ represents seconds.

hh A numeric literal or the name of a binary fullword defined in the Working-Storage or Linkage Section that represents hours.

mins A numeric literal or the name of a binary fullword defined in the Working-Storage or Linkage Section that represents minutes.

secs A numeric literal or the name of a binary fullword defined in the Working-Storage or Linkage Section that represents seconds.

**Data types**

The value or data field you specify for an argument must match the data type required by the option. Usually, one of four specific types is required:

- **binary halfword** PIC S9(4) COMP or PIC X(2)
- **binary fullword** PIC S9(8) COMP or PIC X(4)
- **binary doubleword** PIC 9(18) COMP or PIC X(9)
- **character string** PIC X(n)

For binary halfwords, fullwords, and doublewords, don’t use PIC X(2), PIC X(4), or PIC X(9) if you need to perform arithmetic on the values the fields contain. If an option doesn’t specify a particular data type, you can use any group or elementary item.
RESP and RESP2 options

You can use the RESP and RESP2 options on any CICS command to test if any exceptional conditions were raised during the execution of the command. The RESP option holds the condition raised and the RESP2 option can help determine why the condition was raised. Unit 8 includes the RESP conditions that can be raised for each command, but I didn’t include any of the RESP2 conditions because they aren’t used very often. For a full listing of the RESP2 codes available for each CICS command, see the IBM CICS Application Programming Reference manual.

Syntax

[ RESP(data-value) [ RESP2(data-value) ] ]

Explanation

RESP A binary fullword (PIC S9(8) COMP) field that receives any exceptional condition that’s raised during the execution of a CICS command. You should include this parameter in most of the CICS commands you code. To test the condition, use the keyword DFHRESP.

RESP2 A binary fullword (PIC S9(8) COMP) field that further describes the condition raised in the RESP parameter. Unlike the RESP option, RESP2 values have no associated symbolic names, and you can’t use the DFHRESP keyword to test the value.

Coding example

EXEC CICS
   READ FILE('ACCOUNT')
      INTO(ACCOUNT-RECORD)
      RIDFLD(ACCOUNT-NUMBER)
      RESP(RESPONSE-CODE)
END-EXEC.

   IF RESPONSE-CODE = DFHRESP(NORMAL)
      MOVE 'Y' TO RECORD-FOUND-SW
   ELSE
      IF RESPONSE-CODE = DFHRESP(NOTFND)
         MOVE 'N' TO RECORD-FOUND-SW
      ELSE
         PERFORM 9999-TERMINATE-PROGRAM
      END-IF
   END-IF.

## Commands by function

Unit 8 presents the CICS commands in alphabetical order because that’s the easiest way to look up a specific command. Occasionally, though, you may want to view the CICS commands that are related by function. So the following lists all of the CICS commands in Unit 8 by function group.

<table>
<thead>
<tr>
<th>Abend support</th>
<th>Console support</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABEND</td>
<td>WRITE OPERATOR</td>
</tr>
<tr>
<td>HANDLE ABEND</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>APPC mapped conversation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOCATE</td>
<td></td>
</tr>
<tr>
<td>CONNECT PROCESS</td>
<td></td>
</tr>
<tr>
<td>CONVERSE</td>
<td></td>
</tr>
<tr>
<td>EXTRACT ATTRIBUTES</td>
<td></td>
</tr>
<tr>
<td>EXTRACT PROCESS</td>
<td></td>
</tr>
<tr>
<td>FREE</td>
<td></td>
</tr>
<tr>
<td>ISSUE ABEND</td>
<td></td>
</tr>
<tr>
<td>ISSUE CONFIRMATION</td>
<td></td>
</tr>
<tr>
<td>ISSUE ERROR</td>
<td></td>
</tr>
<tr>
<td>ISSUE PREPARE</td>
<td></td>
</tr>
<tr>
<td>ISSUE SIGNAL</td>
<td></td>
</tr>
<tr>
<td>RECEIVE</td>
<td></td>
</tr>
<tr>
<td>SEND</td>
<td></td>
</tr>
<tr>
<td>WAIT CONVID</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Authentication</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE PASSWORD</td>
<td></td>
</tr>
<tr>
<td>SIGNOFF</td>
<td></td>
</tr>
<tr>
<td>SIGNON</td>
<td></td>
</tr>
<tr>
<td>VERIFY PASSWORD</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BMS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PURGE MESSAGE</td>
<td></td>
</tr>
<tr>
<td>RECEIVE MAP</td>
<td></td>
</tr>
<tr>
<td>ROUTE</td>
<td></td>
</tr>
<tr>
<td>SEND CONTROL</td>
<td></td>
</tr>
<tr>
<td>SEND MAP</td>
<td></td>
</tr>
<tr>
<td>SEND PAGE</td>
<td></td>
</tr>
<tr>
<td>SEND TEXT</td>
<td></td>
</tr>
<tr>
<td>SEND TEXT NOEDIT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Built-in functions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BIF DEEDIT</td>
<td></td>
</tr>
</tbody>
</table>
Interval control
  ASKTIME
  CANCEL
  DELAY
  FORMATTIME
  POST
  RETRIEVE
  START
  WAIT EVENT

Journaling
  JOURNAL
  WAIT JOURNALNAME
  WAIT JOURNALNUM
  WRITE JOURNALNAME
  WRITE JOURNALNUM

Named counter server
  DEFINE COUNTER/DCOUNTER
  DELETE COUNTER/DCOUNTER
  GET COUNTER/DCOUNTER
  QUERY COUNTER/DCOUNTER
  REWIND COUNTER/DCOUNTER
  UPDATE COUNTER/DCOUNTER

Program control
  LINK
  LOAD
  RELEASE
  RETURN
  XCTL

Scheduling
  START ATTACH
  START BREXIT

Security
  QUERY SECURITY

Storage control
  FREEMAIN
  GETMAIN

Syncpoint
  SYNCPOINT

Task control
  CHANGE TASK
  DEQ
  ENQ
  SUSPEND

TCP/IP services
  EXTRACT CERTIFICATE
  EXTRACT TCPIP

Temporary storage control
  DELETEQ TS
  READQ TS
  WRITEQ TS

Terminal control
  ALLOCATE
  BUILD ATTACH
  CONVERSE
  EXTRACT ATTACH
  EXTRACT ATTRIBUTES
  FREE
  HANDLE AID
  ISSUE COPY
  ISSUE DISCONNECT
  ISSUE ERASEAUP
  ISSUE PRINT
  RECEIVE
  SEND

Transient data
  DELETEQ TD
  READQ TD
  WRITEQ TD

Web services
  WEB ENDBROWSE FORMFIELD
  WEB ENDBROWSE HTTPHEADER
  WEB EXTRACT
  WEB READ FORMFIELD
  WEB READ HTTPHEADER
  WEB READNEXT FORMFIELD
  WEB READNEXT HTTPHEADER
  WEB RECEIVE
  WEB RETRIEVE
  WEB SEND
  WEB STARTBROWSE FORMFIELD
  WEB STARTBROWSE HTTPHEADER
  WEB WRITE HTTPHEADER
CICS commands

This unit provides a complete reference to all of the CICS commands you’re likely to use in a command-level program. The commands are listed in alphabetical order to make each entry easy to locate. For each command, you’ll find:

- a brief statement of the command’s function
- the command’s syntax
- an explanation of each of the command’s options
- a listing and explanation of the exceptional conditions that can be raised by the command
- notes and tips that will help you better understand how the command works or help you use the command more effectively
- one or more real-life examples of how to use the command

Note that for any CICS command, you can code the RESP and RESP2 options to handle any exceptional conditions that arise. If you’re maintaining older programs, you may also encounter the HANDLE CONDITION command being used for exception handling. However, using the RESP options results in code that’s simpler, easier to manage, and more efficient at run-time, so that’s the approach that’s recommended today. An overview of exception handling using RESP and RESP2 is given in Unit 2, and the syntax details are given in Unit 7.

A second option that can be coded on any command is the NOHANDLE option. It tells CICS to ignore any HANDLE CONDITION command in effect and to skip the default action if an exceptional condition is raised. Because RESP, RESP2, and NOHANDLE can be coded for any CICS command, they aren’t listed in the command syntax in this unit.
The DEFINE COUNTER command

Function

Named counter server. The DEFINE COUNTER command lets you define a new named counter in a named counter pool of the coupling facility.

Syntax

```
EXEC CICS
  DEFINE { COUNTER(name) | DCOUNTER(name) }
  [ POOL(name) ]
  [ VALUE(data-value) ]
  [ MINIMUM(data-value) ]
  [ MAXIMUM(data-value) ]
END-EXEC.
```

Options

- **COUNTER** Specifies the 16-character name of the named counter to be created. All the value fields for this counter will then be handled as signed binary fullwords (PIC S9(8) COMP).
- **DCOUNTER** Specifies the 16-character name of the named counter to be created. All the value fields for this counter will then be handled as unsigned binary doublewords (PIC 9(18) COMP).
- **POOL** Specifies the 8-character name of the pool in which the named counter is to be created. If there’s no matching entry in the DFHNCOPT options table, the default named counter pool on the NCPLDFT system initialization parameter is used.
- **VALUE** Specifies a signed binary fullword (PIC S9(8) COMP), unsigned binary doubleword (PIC 9(18) COMP), or literal value that indicates the starting value for the counter. You can specify a number that is equal to, or greater than, the minimum value specified and up to the maximum value plus 1. If you omit this option and the MINIMUM option, the counter is created with an initial value of zero.
- **MINIMUM** Specifies a signed binary fullword (PIC S9(8) COMP), unsigned binary doubleword (PIC 9(18) COMP), or literal value that indicates the minimum value for the counter. The counter is reset to this value after a REWIND COUNTER command. If MINIMUM is omitted, the default counter minimum is set to LOW-VALUE (hex zeros). However, if MINIMUM is coded, the VALUE option must also be coded or the CICS translator will issue an error.
- **MAXIMUM** Specifies a signed binary fullword (PIC S9(8) COMP), unsigned binary doubleword (PIC 9(18) COMP), or literal value that indicates the maximum value for the counter. Once the counter reaches this number, it must be reset by the REWIND COUNTER command. If MAXIMUM is omitted, the default counter maximum is set to HIGH-VALUE (hex FF).
Exceptional conditions

INVREQ The statement was coded improperly or there was a problem defining the named counter in the coupling facility. The default action for this condition is to terminate the task.

Notes and tips

- You use the DEFINE COUNTER command to create a named counter that generates unique sequence numbers. You can then use these values for functions such as assigning control numbers (like customer, account, or invoice numbers) by using the GET COUNTER command.

- The named counter facility is designed to run in a Parallel Sysplex environment. That means that the facility is controlled by a named counter server, allowing multiple regions (CICS or non-CICS) to draw from the same counter.

- Before the named counter facility was available, in order to share a set of sequence numbers in a CICS application, you had to use either a shared CICS data table or a CICS common work area (CWA) to store a number that was updated by each application. The problem with the shared CICS data table is that all of the CICS regions have to reside on the same MVS image. And the problem with the CWA is that it can only be used within the same CICS region.

- If you use a field, not a literal value, to assign a COUNTER or DCOUNTER name, make sure that the field is padded with trailing spaces if the name is less than 16 characters long.

Coding example

The following example shows a DEFINE COUNTER command that defines a counter named ORDERINV that’s used to assign invoice numbers. The counter is created in a pool area named MMA, and its initial value is 100 with a maximum value of 999999.

```
EXEC CICS
    DEFINE COUNTER('ORDERINV')
        POOL('MMA')
        VALUE(100)
        MINIMUM(100)
        MAXIMUM(999999)
END-EXEC.
```
The DOCUMENT CREATE command

Function

Document services. (TS 1.3 and later) The DOCUMENT CREATE command is typically used to create HTML documents and special forms. The document created can be empty or its contents can be based on an existing document.

Syntax

```
EXEC CICS
   DOCUMENT CREATE   DOCTOKEN(data-area)
      [ { FROM(data-area) LENGTH(data-value) } ]
      [ { TEXT(data-area) LENGTH(data-value) } ]
      [ { BINARY(data-area) LENGTH(data-value) } ]
      [ FROMDOC(data-area) ]
      [ TEMPLATE(name) ]
      [ SYMBOLLIST(data-area) LISTLENGTH(data-value) ]
      [ DELIMITER(data-value) ]
      [ UNESCAPED ]
      [ DOCSIZE(data-area) ]
      [ HOSTCODEPAGE(name) ]
END-EXEC.
```

Options

- **DOCTOKEN**: Specifies the 16-character symbolic name of the document to be created. CICS generates this name when the document is created. Other DOCUMENT commands can then refer to the document by this name.
- **FROM**: Specifies the data area that’s to be used to create the document. This data area can be a template or an existing document that’s been retrieved.
- **TEXT**: Specifies a character string data area that’s used to create the document. No attempt is made to parse the data for symbol substitution.
- **BINARY**: Specifies a binary data area that’s used to create the document. This option lets you insert a block of data that won’t be converted to the client’s code page when the data is sent, allowing embedded code to be passed to the new document.
- **LENGTH**: Specifies a binary fullword (PIC S9(8) COMP) that contains the length of the data area used by the FROM, TEXT, or BINARY options to create the document.
- **FROMDOC**: Specifies the DOCTOKEN name of another document whose contents are to be copied to the new document being created.
- **TEMPLATE**: Specifies the 48-character name of a template defined to CICS through the RDO facility. The contents of the template is copied to the new document being created.
SYMBOLLIST Specifies a field that contains a symbol list. A symbol list is a character string that consists of one or more symbol definitions separated by ampersands. Each symbol definition consists of a name, an equals sign, and a value. Example: tranid=INV1&user=MM01&orderno=123456. During document creation, the values in the symbol list are substituted for the corresponding names (symbols) in the document contents.

LISTLENGTH Specifies a binary fullword (PIC S9(8) COMP) that contains the length of the SYMBOLLIST.

DELIMITER (TS 2.1 and later) Specifies a 1-character field or literal value used to delimit symbol definitions in the SYMBOLLIST. If not specified, the value defaults to an ampersand.

UNESCAPED (TS 2.1 and later) Prevents CICS from escaping symbol values contained in the SYMBOLLIST. If used, plus signs are not converted to spaces, and sequences in the format $%nn$ are not converted to single character values (where $nn$ is the ASCII value for the character).

DOCSIZE Specifies a binary fullword (PIC S9(8) COMP) that’s updated with the current size of the document in bytes.

HOSTCODEPAGE Specifies the 8-character name of the host codepage to be used for the data being added to the document. HOSTCODEPAGE can only be used with the TEXT, TEMPLATE, and SYMBOLLIST options.

### Exceptional conditions

*Note: The default action for these conditions is to terminate the task.*

- **INVREQ** The document or template specified in the FROM option is not in a valid format.
- **NOTFND** The document or template specified in the FROMDOC or TEMPLATE option can’t be found or is named incorrectly.
- **SYMBOLERR** A symbol specified in the symbol list doesn’t conform to the naming rules for symbols.
- **TEMPLATERR** An invalid #set, #include, or #echo command was encountered while processing the specified template.

### Notes and tips

- The DOCUMENT commands allow you to create and manage HTML pages that can then be sent to a web browser through CICS’s web support facility. Prior to the introduction of the DOCUMENT commands in CICS TS 1.3, an HTML template manager was needed. Although it can still be used, we recommend you use the DOCUMENT commands instead.

- The DOCUMENT CREATE command is the first step in creating a document. The document created can be empty or based on another document, template, or data string.
A template contains generalized information that can be used as the basis for many different documents. When you use the TEMPLATE option, you can code the SYMBOOLLIST option to change any of the predefined symbols in the template to values you specify. This allows you to customize your document with specific information related to your task. Any template referenced must first be defined to CICS through the RDO facility.

- Remember to adhere to HTML coding standards when formatting or adding symbols to an HTML document.

- Some characters, such as the ampersand (&), have special meaning in HTML. So to avoid confusion when a user enters a character like that, the HTTP client escapes the character. That means the character is converted to the format %nn, where nn is the ASCII value of the character. Unescaping means a character in that format is not converted to a single character but is transmitted as is.

- A host code page and a client code page are used to translate data to the format that’s used on the server and the client, respectively. In an IBM environment, usually the host uses EBCDIC code while the clients use ASCII. Specifying what the host and client code pages are if they’re not standard EBCDIC and ASCII will ensure that the information being transmitted will be translated correctly.
Coding example

The following code shows how the DOCUMENT CREATE command is used to create an HTML document that displays an order and confirmation number. The template used to create the document is called ORDERCONFIRM, and the SYMBOLLIST option passes information that’s specific to the order. The WEB SEND command then passes the HTML page back out to the web server for delivery to the client browser.

```
WORKING-STORAGE SECTION.

01  CURRORDER       PIC X(16) VALUE SPACES.
01  STATUS-MESSAGE  PIC X(10) VALUE 'OK'.

01  CONFIRM-STRING.
   05  FILLER      PIC X(08) VALUE 'order_no'.
   05  FILLER      PIC X(01) VALUE '='.
   05  CONF-ODR-NO PIC 9(06) VALUE ZERO.
   05  FILLER      PIC X(01) VALUE '&'.
   05  FILLER      PIC X(10) VALUE 'confirm_no'.
   05  FILLER      PIC X(01) VALUE '='.
   05  CONF-NO     PIC X(09) VALUE SPACE.

01  CODEPAGE-INFO.
   05  CODEPAGE-EBCDIC  PIC X(08) VALUE '037'.
   05  CODEPAGE-ASCII   PIC X(40) VALUE 'iso-8859-1'.

PROCEDURE DIVISION.

5000-SEND-CONFIRMATION.

EXEC CICS
   DOCUMENT CREATE
      DOCTOKEN(CURRORDER)
      TEMPLATE('ORDERCONFIRM')
      SYMBOLLIST(CONFIRM-STRING)
      LISTLENGTH(LENGTH OF CONFIRM-STRING)
      HOSTCODEPAGE(CODEPAGE-EBCDIC)
END-EXEC.

EXEC CICS
   WEB SEND
      DOCTOKEN(CURRORDER)
      STATUSCODE(200)
      STATUS(MESSAGE)
      CLNTCODEPAGE(CODEPAGE-ASCII)
END-EXEC.
```
The ENQ command

Function

Task control. The ENQ command reserves a user-defined resource for exclusive use by your task. Any other task that issues an ENQ command for the same resource will be suspended until your task ends or issues a DEQ command for the resource.

Syntax

```c
EXEC CICS
   ENQ   RESOURCE(data-area)
   [ LENGTH(data-value) ]
   [ UOW | TASK | MAXLIFETIME(data-area) ]
   [ NOSUSPEND ]
END-EXEC.
```

Options

**RESOURCE**
Identifies the resource to be reserved. If LENGTH is also specified, the character string (up to 255 bytes) contained in the data area is used to identify the resource; if LENGTH is omitted, the address of the data area identifies the resource.

**LENGTH**
Specifies a binary halfword (PIC S9(4) COMP) or literal value that indicates the length (up to 255 bytes) of the character string specified in the RESOURCE option.

**UOW**
Specifies that the enqueue should be held until the end of the current unit of work (the default). For compatibility with previous releases of CICS/ESA, LUW can also be used.

**TASK**
Specifies that the enqueue should be held until the end of the current task.

**MAXLIFETIME**
Specifies a binary fullword (PIC S9(8) COMP) that indicates the duration of the enqueued resource. The field’s value can be set to either DFHVALUE(UOW) or DFHVALUE(TASK).

**NOSUSPEND**
Indicates that if the resource is already reserved, control is returned immediately to your program at the point following the ENQ command.

Exceptional conditions

*Note: The default action for all of these conditions except ENQBUSY is to terminate the task. The default action for the ENQBUSY condition is to suspend the task.*

**ENQBUSY**
Indicates that another task has already issued an ENQ command naming the resource you specified.

**INVREQ**
The MAXLIFETIME value is incorrect.

**LENGERR**
The length value is not in the range of 1 to 255.
Notes and tips

- The ENQ/DEQ facility is useful for single threading access to resources that you don’t want to be shared: printers, destinations, temporary storage queues, etc.
- The ENQ/DEQ facility will work properly only if all tasks that use a particular resource issue ENQ commands and identify the resource in the same way. The ENQ/DEQ facility does nothing to prevent programs that don’t issue an ENQ command from accessing the resource.
- Most installations have a standard that specifies how resources are named. Find out what that standard is and be sure to follow it.
- If the LENGTH option is specified in the ENQ command, it must also be specified in the DEQ command for the same resource and the value must match.
- ENQBUSY is one of the few exceptional conditions that doesn’t cause your task to be terminated. Instead, its action depends on whether you handle the condition (using the RESP option or a HANDLE CONDITION command) and whether you specify NOSUSPEND on the ENQ command. If you do neither, ENQBUSY simply causes your task to be suspended until the resource becomes available; then, control returns to your program at the first statement following the ENQ command. If your program handles the ENQBUSY condition, the specified error processing is done, and the program must issue the ENQ command again to reserve the resource. If you specify NOSUSPEND, control is returned to the statement following the ENQ command if the resource is unavailable; again, you’ll have to issue another ENQ command to reserve the resource later.
Coding example

This example shows how to issue an ENQ command to reserve a printer so the program can send output to it. The character string in the RESOURCE option is the printer-id. Thus, any other task that issues an ENQ command using the same printer-id will be suspended until this program issues the DEQ command. Without this command, other programs could access the printer at the same time, so the output might be interspersed with data from other applications.

```
01 PRINTER-ID PIC X(04) VALUE 'L86P'.
```

```
PROCEDURE DIVISION.

0000-PRODUCE-INVENTORY-LISTING.

EXEC CICS
  ENQ RESOURCE(PRINTER-ID)
  LENGTH(4)
END-EXEC.

PERFORM 2000-PRINT-INVENTORY-LINE
  UNTIL END-OF-REPORT.

PERFORM 3000-PRINT-TOTAL-LINE.

EXEC CICS
  DEQ RESOURCE(PRINTER-ID)
  LENGTH(4)
END-EXEC.
```
The LINK command

Function

Program control. The LINK command invokes a program, which causes the program to be loaded into storage if necessary, and passes data to the invoked program if needed. When the invoked program ends, control is returned to the statement following the LINK command in the invoking program. In some cases, the invoked program may reside on another system.

Syntax

```plaintext
EXEC CICS LINK   PROGRAM(name)
                [ COMMAREA(data-area) ]
                [ LENGTH(data-value) ]
                [ DATALENGTH(data-value) ]
                [ INPUTMSG(data-area) ]
                [ INPUTMSGLEN(data-value) ]
                [ SYSID(systemname) ]
                [ SYNCRETURN ]
                [ TRANSID(name) ]
END-EXEC.
```

Options

PROGRAM Specify the 1- to 8-character name of the program to be invoked. If this name is not defined in CICS and AUTOINSTALL is active, CICS will supply a definition for the program.

COMMAREA Specify a data area that’s passed to the invoked program as a communication area. The invoked program accesses the communication area via its DFHCOMMAREA field, which is addressed to the data area specified in the invoking program. In other words, the invoked program accesses the communication area in the same storage locations as the invoking program; the communication area is not copied to another area of storage. (This works differently than it does for an XCTL or RETURN command; see the descriptions of those commands for details.)

LENGTH Specify a binary halfword (PIC S9(4) COMP) or literal value that indicates the length of the data area specified in the COMMAREA option. This value can’t exceed 32500 bytes if the COMMAREA is to be passed between two CICS systems.

DATALENGTH Specify a binary halfword (PIC S9(4) COMP) or literal value that indicates the length of the data to be sent from the area specified in the COMMAREA option. The value may be less than the total length of this area.

INPUTMSG Specify a data area that will be used to provide input for a RECEIVE command issued by the invoked program. This option can’t be used at the same time as DATALENGTH.
INPUTMSGLEN Specifies a binary halfword (PIC S9(4) COMP) or literal value that indicates the length of the INPUTMSG field.

SYSID Specifies the name of the remote system where the specified program resides. If omitted, the program’s resource definition is used to determine the location of the program.

SYNCONRETURN Indicates that the server program should perform a syncpoint when it returns control to the client program. If omitted, the syncpoint is taken when the client program ends.

TRANSID Specifies the trans-id that the remote system should use to run the invoked program. If omitted, CSMI, CPMI, or CVMI is used.

Note: If you specify INPUTMSG, you cannot specify SYSID, SYNCONRETURN, or TRANSID.

**Exceptional conditions**

*Note: The default action for these conditions is to terminate the task.*

- INVREQ INPUTMSG, SYNCONRETURN, or TRANSID is used improperly.
- LENGERR A length error occurred.
- NOTAUTH A resource security check has failed on the program named in the PROGRAM option.
- PGMIDERR The program is not defined in the Processing Program Table (PPT).
- ROLLEDBACK SYNCONRETURN was specified, and the linked-to program could not successfully take a syncpoint. The linked-to program issued a rollback.
- SYSIDERR The remote system could not be found.
- TERMERR A conversation error occurred.

**Notes and tips**

* The LINK command invokes a CICS program as if it were a subprogram. You can achieve a similar result by using a COBOL Call statement. Although the Call statement is faster and more efficient (because CICS isn’t involved), the LINK command has some distinct advantages: It automatically provides addressability to the EIB, and you can invoke a program on either a local or remote system. A COBOL Call statement can only invoke a program on a local system.

* When you invoke a program using LINK, CICS keeps the invoking program in virtual storage while the linked program executes. In contrast, the XCTL command, which also transfers control to another program, doesn’t use a return mechanism, so CICS is free to release resources used by the invoking program. Despite that, both XCTL and LINK consume CICS resources. So the choice of one over another should be based on transaction design and not on perceived performance differences.
• If application processing can continue without the linked program, you should handle the PGMIDERR and NOTAUTH conditions. If the program is critical to the application, though, there’s little point in handling those conditions.

• The INPUTMSG option is provided to pass data to programs that were originally designed to receive data from the terminal. Data placed in the INPUTMSG area is received by the linked-to program’s first RECEIVE command.

• The SYSID, SYNCONRETURN, and TRANSID options were introduced with CICS/ESA 3.3 to support Distributed Program Link (DPL). In most cases, all three can be omitted. Then, the SYSID is specified by the program’s resource definition. If the linked-to program updates recoverable resources and the linking program does no updates after issuing the LINK, you can save some communication time by using the SYNCONRETURN option. If you do, however, be sure to test for the ROLLEDBACK condition.

Coding example

This example invokes a program named MMIN2010, passing it a communication area named NEXT-INVOICE-NUMBER.

```
5000-GET-NEXT-INVOICE-NUMBER.
*  
  EXEC CICS
  LINK PROGRAM('MMIN2010')
  COMMAREA(NEXT-INVOICE-NUMBER)
END-EXEC.
```
**The READNEXT command**

**Function**

*File control.* The READNEXT command retrieves the next sequential record from a file during a browse operation. The file can be a VSAM KSDS, ESDS, RRDS, or path.

**Syntax**

```cics
EXEC CICS READNEXT FILE(filename)
{ INTO(data-area) | SET(pointer-ref) }
RIDFLD(data-area)
[ { LENGTH(data-area) }
{ SYSID(systemname) LENGTH(data-area) } ]
[ KEYLENGTH(data-value) ]
[ RBA | RRN ]
[ REQID(data-value) ]
[ { UPDATE
   [ TOKEN(data-area) ] }
{ CONSISTENT }
{ REPEATABLE }
{ UNCOMMITTED } ]
[ NOSUSPEND ]
END-EXEC.
```

**Options**

- **FILE** Specifies the 1- to 8-character name of the data set that contains the record to be read.
- **INTO** Specifies the data area that will contain the record being read.
- **SET** Specifies the data area that will contain the address of the retrieved record.
- **RIDFLD** Specifies a data area that identifies the record to be retrieved. The content of the RIDFLD field depends on whether RBA or RRN is specified; if neither is specified, the RIDFLD field contains a key for VSAM KSDS or path retrieval. For a generic browse, the RIDFLD field must still be as long as the file’s defined key length. Then, when the READNEXT command finishes, CICS puts the complete key value in the RIDFLD field. Normally, you should leave the contents of the RIDFLD field unchanged during a browse operation. If you change the field before you issue a READNEXT command, the browse is restarted from the new location.
- **LENGTH** Specifies a binary halfword (PIC S9(4) COMP) that contains the length of the record. On entry, the data area indicates the size of the INTO data area if INTO is specified. On exit, the data area contains the size of the record retrieved. Required if INTO is specified and the file has variable-length records; optional for SET. Also required when SYSID is specified.
SYSID Specifies the 1- to 4-character name of a remote system that contains the file.

KEYLENGTH Specifies a binary halfword (PIC S9(4) COMP) or literal value that indicates the length of the key. If you specified GENERIC on the STARTBR or RESETBR command, you can use the KEYLENGTH option to change the length of the generic key. In that case, CICS repositions the browse using the new generic key. Also used when the SYSID option is specified. Not valid if RBA or RRN is specified.

RBA Specifies that the RIDFLD field is a relative byte address (RBA) for a VSAM KSDS or ESDS. An RBA is a binary fullword (PIC S9(8) COMP).

RRN Specifies that the RIDFLD is a relative record number (RRN) for a VSAM RRDS. An RRN is a binary fullword (PIC S9(8) COMP). The RRN of the first record in an RRDS is 1.

REQID Specifies a binary halfword (PIC S9(4) COMP) or literal value that identifies the browse operation; used only when your program controls two or more browse operations at the same time. For each I/O command that’s part of the same browse operation, specify the same REQID value.

UPDATE (TS 1.1 and later) (RLS only) Specifies that you intend to update the record by rewriting or deleting it. The record is held under exclusive control by your task, and no other task can access it until: (1) you issue a REWRITE, DELETE, or UNLOCK command to release the record or (2) your task ends. If the file is recoverable, the record is held until the task ends or you issue a SYNCPOINT command, even if you issue a REWRITE, DELETE, or UNLOCK command. If you specify the UPDATE option, you must also specify the TOKEN option.

TOKEN (TS 1.1 and later) (RLS only) Specifies a binary fullword (PIC S9(8) COMP) that returns a unique identifier for a READNEXT UPDATE request. This identifier can then be used to associate the record retrieved with a subsequent REWRITE, DELETE, or UNLOCK command. The UPDATE option is assumed if TOKEN is specified.

CONSISTENT (RLS only) Specifies that the record is to be read with a level of read integrity provided by a VSAM shared lock that lasts until the record is read (that is, until it’s returned to your program). If the record is locked by another task, the READ request waits until the lock is released.

REPEATABLE (RLS only) Specifies that the record is to be read with a level of read integrity provided by a VSAM shared lock that lasts until the unit of work that contains the READ request is ended. If the record is locked by another task, the READ request waits until the lock is released.

UNCOMMITTED (RLS only) Specifies that the record is to be read with no read integrity.

NOSUSPEND (RLS only) Specifies that the READNEXT request will not wait if the record is held under exclusive control (locked) by another task.
Exceptional conditions

Note: The default action for these conditions is to terminate the task.

DUPKEY Occurs only when retrieving records via an alternate index (path) that allows duplicate keys; indicates that at least one more record with the specified key exists. To retrieve all of the records with the same key value, issue successive READNEXT commands; DUPKEY will be raised for each record except the last.

ENDFILE Occurs when there are no more records to be retrieved.

FILENOTFOUND The data set name specified in the FILE option isn’t defined in the File Control Table (FCT).

ILLOGIC A serious VSAM error occurred.

INVREQ A browse operation has not been properly started by a STARTBR command, the meaning of the RIDFLD option (key, RBA, or RRN) was changed during the browse, or the KEYLENGTH value is incorrect.

IOERR An I/O error occurred.

ISCINVREQ An undeterminable error occurred on the remote system specified in the SYSID option.

LENGERR The length of the record retrieved exceeds the length specified in the LENGTH option.

LOADING (TS 1.1 and later) The READNEXT request was issued against a data table that is still being loaded into the CICS system.

LOCKED A READNEXT UPDATE was issued against a record that is currently locked.

NOTAUTH The transaction’s PCT entry specified that resource security checking should be done, and the operator is not authorized to access the data set.

NOTFND The specified record could not be located.

NOTOPEN The file is not open.

RECORDBUSY (TS 1.1 and later) The NOSUSPEND option was specified on a record held under exclusive control by another task.

SYSIDERR The system identified by SYSID could not be located or accessed.

Notes and tips

- Before you can issue a READNEXT command, you must begin a browse operation by issuing a STARTBR command. Browsing is relatively inefficient because a VSAM string is held for the duration of the browse. (A string is required for each concurrent access to a VSAM file, so if 10 strings are specified for a file, 10 simultaneous accesses are permitted.) Because of this inefficiency, you may want to minimize the duration of a browse so you don’t tie up a string any longer than is needed.
• Record-level sharing (RLS) is a VSAM feature available through the SMS facility. It allows files to be shared with update capability between applications and across CICS regions. When you open a file in RLS mode, locking takes place at the record level instead of the control-interval level. This reduces the possibility of deadlocks.

• If you have RLS active on your system and you’re working in CICS TS 1.1 or higher, you can invoke the READNEXT command with the UPDATE option. This feature allows you to update or delete the record retrieved. Without it, you must first end the browse with an ENDBR command and then issue a READ UPDATE command for the record you want to update or delete.

• To browse a file via an alternate path, specify the path name in the FILE option. Be aware that if the alternate index allows duplicate keys, the DUPKEY condition will be raised if there’s more than one record with the same alternate key value. So be sure to check for the DUPKEY condition.

• When you retrieve records with duplicate keys, the records are presented in the sequence they were created. If the alternate indexes were recently rebuilt, that will be in prime key sequence. But don’t count on it. (Actually, if the alternate index isn’t upgradable, any duplicates will always be in prime key sequence since the alternate index entries can be created only by rebuilding the index.)

• The only exceptional condition you normally need to worry about for the READNEXT command is ENDFILE. If you’re browsing via a path that allows duplicate keys, you should also check for DUPKEY.
Coding example (base cluster)

This example shows how to retrieve records during a browse operation. The records are retrieved directly from the base cluster of a VSAM KSDS via the primary key. Several program modules are shown to indicate the program logic necessary to invoke the browse module repeatedly. (The code for module 1100-START-ACCOUNT-BROWSE is shown in the coding example for the STARTBR command later in this unit.)

```
MOVE LOW-VALUE TO AR-ACCOUNT-NUMBER.
PERFORM 1100-START-ACCOUNT-BROWSE.
PERFORM 2000-PROCESS-ACCOUNT-RECORD
   UNTIL END-OF-BROWSE.

* 2000-PROCESS-ACCOUNT-RECORD.
* PERFORM 2100-READ-NEXT-ACCOUNT-RECORD.
IF NOT END-OF-BROWSE
   .
*
2100-READ-NEXT-ACCOUNT-RECORD.
*
EXEC CICS
   READNEXT FILE('ACCOUNT')
      INTO(ACCOUNT-RECORD)
      RIDFLD(AR-ACCOUNT-NUMBER)
      RESP(RESPONSE-CODE)
END-EXEC.

IF RESPONSE-CODE = DFHRESP(ENDFILE)
   MOVE 'Y' TO END-OF-BROWSE-SW
ELSE
   IF RESPONSE-CODE NOT = DFHRESP(NORMAL)
      PERFORM 9999-TERMINATE-PROGRAM
   END-IF
END-IF.
```
Coding example (alternate index)

This example shows how to retrieve duplicate key records via an alternate index. Here, module 1800 retrieves up to 10 invoice records by customer number, which is an alternate key for the file. The DUPKEY condition indicates whether more invoices are available for the customer. As soon as DUPKEY is not detected, N is moved to MORE-INVOICES-SW.

```
1800-GET-INVOICE-RECORDS.
  *
  PERFORM 1810-START-INVOICE-BROWSE.
  PERFORM 1820-FORMAT-INVOICE-LINE
       VARYING INVOICE-SUB FROM 1 BY 1
       UNTIL INVOICE-SUB > 10.
  *
  1820-FORMAT-INVOICE-LINE.
  *
  IF MORE-INVOICES
     PERFORM 1830-READ-INVOICE-RECORD
     MOVE INV-INVOICE-NUMBER
           TO IM-D-INVOICE-NUMBER(INVOICE-SUB)
     MOVE INV-PO-NUMBER
           TO IM-D-PO-NUMBER(INVOICE-SUB)
     MOVE INV-INVOICE-DATE
           TO IM-D-INVOICE-DATE(INVOICE-SUB)
     MOVE INV-INVOICE-TOTAL
           TO IM-D-INVOICE-TOTAL(INVOICE-SUB)
     ELSE
     MOVE SPACE TO IM-D-INVOICE-LINE(INVOICE-SUB)
     END-IF.
  *
1830-READ-INVOICE-RECORD.
  *
  EXEC CICS
     READNEXT FILE('INVPATH')
       INTO(INVOICE-RECORD)
       RDFLD(CM-CUSTOMER-NUMBER)
       RESP(RESPONSE-CODE)
  END-EXEC.
  *
  IF RESPONSE-CODE = DFHRESP(NORMAL)
     MOVE 'N' TO MORE-INVOICES-SW
  ELSE
     IF RESPONSE-CODE NOT = DFHRESP(DUPKEY)
        PERFORM 9999-TERMINATE-PROGRAM
     END-IF
  END-IF.
```
The SEND MAP command

Function

Basic Mapping Support. The SEND MAP command lets you send data to a terminal, mapping it according to the specifications in a BMS map definition.

Syntax

```plaintext
EXEC CICS
SEND MAP(name)
[ MAPSET(name) ]
[ FROM(data-area) ]
[ LENGTH(data-value) ]
[ DATAONLY | MAPONLY ]
[ ERASEAUP | ERASE ]
[ ALARM ]
[ FREEKB ]
[ FRSET ]
[ CURSOR [(data-value)] ]
[ PRINT ]
[ FORMFEED ]
[ NLEOM ]
[ ACCUM ]
[ PAGING | SET(pointer-ref) | TERMINAL [WAIT] [ LAST] ]
[ L40 | L64 | L80 | HONEOM ]
[ REQID(name) ]
[ NOFLUSH ]
END-EXEC.
```

Options

MAP Specifies the 1- to 7-character name of the map to be used to map the output data.

MAPSET Specifies the 1- to 8-character name of the mapset that contains the map. If omitted, the map name is used. This name must be defined in the Processing Program Table (PPT).

FROM Specifies the data area that contains the data to be mapped.

LENGTH Specifies a binary halfword (PIC S9(4) COMP) or literal value that indicates the length of the data to be mapped. Required only if less than the entire data area specified in the FROM option is to be used.

DATAONLY Data from the FROM area is to be mapped, but not constant data included in the BMS map definition.

MAPONLY Only constant data from the BMS map definition is to be sent; no FROM area is used.

ERASEAUP Erases all of the unprotected fields on the screen.

ERASE Erases the entire display screen. When used with the ACCUM option, ERASE causes the display to be erased as each page is displayed, not as each map is written.
ALARM       Sounds the terminal’s alarm.
FREEKB      Unlocks the terminal’s keyboard.
FRSET       Resets the Modified Data Tag (MDT) bit of each attribute byte to zero.
CURSOR      Specifies a binary halfword (PIC S9(4) COMP) or literal value that indicates the position where the cursor is to be placed. The row and column corresponding to a given cursor position depends on the number of columns in each line. For an 80-column display, column 1 of row 1 is cursor position 0, column 1 of row 2 is cursor position 80, and so on. If you specify CURSOR but omit the data value, the symbolic cursor positioning technique (described in Unit 2) is used.
PRINT       When used with a printer, specifies that the data is to be printed. If PRINT is omitted, the data is sent to the printer but not printed.
FORMFEED    Causes the printer to advance to the top of the next page.
NLEOM       Specifies that BMS is to use new-line (NL) and end-of-message (EM) orders to build the output; should be used for output intended for a printer.
ACCUM       Specifies that this command is part of a message building operation.
PAGING       Specifies that output should be held in temporary storage until it can be delivered to its final destination.
SET         Specifies a pointer to be set to the address of the output data.
TERMINAL    Specifies that output should be sent directly to the terminal. This is the default.
WAIT        Specifies that the task should be suspended until the output operation has completed.
LAST        For logical units only, specifies the last terminal output operation for the task.
L40         Specifies that the maximum line length for printed output is 40 characters.
L64         Specifies that the maximum line length for printed output is 64 characters.
L80         Specifies that the maximum line length for printed output is 80 characters.
HONEOM      Specifies that CICS should honor the printer’s default end-of-margin setting when determining the maximum print line length. This is the default.
REQID       Specifies a two-character name that’s used for message recovery. If omitted, ‘**’ is assumed. All BMS commands for the same logical message must specify the same REQID value.
NOFLUSH     Specifies that the system not clear pages if an OVERFLOW condition occurs.
Exceptional conditions

Note: The default action for all of these conditions except OVERFLOW is to terminate the task. The default action for OVERFLOW is to ignore the condition.

IGREQCD A VTAM error has occurred.
IGREQID The prefix specified in the REQID option is different from the prefix established in a previous REQID.
INVMP$SZ The map is too large for the terminal.
INVREQ The request is not allowed.
OVERFLOW There is not enough room on the screen for the map. If you test the RESP or EIBRESP field to detect this condition, you can issue additional SEND MAP commands to complete the current page by sending a trailer map and/or a header map for the next page.
TSIOERR A temporary storage I/O error has occurred.

Notes and tips

- If you omit the FROM option, BMS adds the letter O to the end of the map name you specify to determine the name of the output data area. So, if the map is named CUSTMAP, the output data area is CUSTMAPO. BMS uses the same convention when it assembles the symbolic map, so if you copy the BMS-generated symbolic mapset into your program, the names will match up. However, you must code FROM if (1) you want to use your own version of the symbolic map or (2) you specify a data name rather than a literal value in the MAP option.

- The DATAONLY and MAPONLY options let you add data to a display. Code DATAONLY when the screen already contains the correct captions, but you want to change the data that’s displayed. Code MAPONLY when you want to display just captions with no data. Omit both if you want data from the symbolic map to be combined with captions coded in the BMS map definition. When you code these options, remember that hex zeros (LOW-VALUE) in the symbolic map are never sent to the terminal. So, moving LOW-VALUE to the symbolic map and issuing a SEND MAP command without coding MAPONLY or DATAONLY is equivalent to issuing the same command with MAPONLY. Usually, you’ll omit both MAPONLY and DATAONLY and move LOW-VALUE to the symbolic map fields that don’t need to be sent to the terminal.

- Although the CURSOR option lets you place the cursor at any screen location, you’ll usually want to use the symbolic cursor positioning technique instead. Symbolic cursor positioning is described in Unit 2.

- If you’re building a logical message, be sure to include the ACCUM and PAGING options and test for the OVERFLOW condition.
This example shows a module that issues one of several varieties of SEND MAP commands depending on the setting of a control flag. In each case, the cursor position is specified using symbolic cursor positioning, as described in Unit 2.

```cics
1400-SEND-CUSTOMER-MAP.
  *
  EVALUATE TRUE
  WHEN SEND-ERASE
    EXEC CICS
      SEND MAP('MNTMAP1')
        MAPSET('MNTSET1')
        FROM(MNTMAP1O)
        ERASE
        CURSOR
    END-EXEC
  WHEN SEND-ERASE-ALARM
    EXEC CICS
      SEND MAP('MNTMAP1')
        MAPSET('MNTSET1')
        FROM(MNTMAP1O)
        ERASE
        ALARM
        CURSOR
    END-EXEC
  WHEN SEND-DATAONLY
    EXEC CICS
      SEND MAP('MNTMAP1')
        MAPSET('MNTSET1')
        FROM(MNTMAP1O)
        DATAONLY
        CURSOR
    END-EXEC
  WHEN SEND-DATAONLY-ALARM
    EXEC CICS
      SEND MAP('MNTMAP1')
        MAPSET('MNTSET1')
        FROM(MNTMAP1O)
        DATAONLY
        ALARM
        CURSOR
    END-EXEC
  END-EVALUATE.
```
Coding example (message building)

This long example shows how to use SEND MAP commands to build message pages, each consisting of multiple maps. Each page consists of three types of maps: a header map (LSTMAP1), which is displayed at the top of each page; a detail map (LSTMAP2), which is displayed several times on each page; and a trailer map (LSTMAP3), which is displayed at the bottom of each page. The OVERFLOW condition is tested to determine when header and trailer maps should be sent.

```plaintext
0000-PRODUCE-PRODUCT-LISTING.
  *
  .
  PERFORM 2230-SEND-HEADER-MAP.
  PERFORM 2000-PRODUCE-PRODUCT-LINE
      UNTIL PRODUCT-EOF.
  EXEC CICS
      SEND PAGE OPERPURGE
      END-EXEC.
  EXEC CICS
      RETURN
      END-EXEC.
  *
  .
  2000-PRODUCE-PRODUCT-LINE.
  *
  PERFORM 2100-READ-PRODUCT-RECORD.
  IF NOT PRODUCT-EOF
     PERFORM 2200-SEND-PRODUCT-LINE.
  *
  .
  2200-SEND-PRODUCT-LINE.
  *
  MOVE PRM-PRODUCT-CODE        TO PCODEO.
  MOVE PRM-PRODUCT-DESCRIPTION TO DESCRO.
  MOVE PRM-UNIT-PRICE          TO UPRICEO.
  MOVE PRM-QUANTITY-ON-HAND    TO ONHANDO.
  PERFORM 2210-SEND-DETAIL-MAP.
  IF PAGE-OVERFLOW
     PERFORM 2220-SEND-TRAILER-MAP
     PERFORM 2230-SEND-HEADER-MAP
     PERFORM 2210-SEND-DETAIL-MAP
     MOVE 'N' TO PAGE-OVERFLOW-SW.
  *
  2210-SEND-DETAIL-MAP.
  *
  EXEC CICS
      SEND MAP('LSTMAP2')
      MAPSET('LSTSET1')
      FROM(LSTMAP2O)
      ACCUM
      PAGING
      ERASE
      RESP(RESPONSE-CODE)
      END-EXEC.
```
IF RESPONSE-CODE = DFHRESP(OVERFLOW)
   MOVE 'Y' TO PAGE-OVERFLOW-SW
ELSE
   IF RESPONSE-CODE NOT = DFHRESP(NORMAL)
      PERFORM 9999-TERMINATE-PROGRAM
   END-IF
END-IF.

* 2220-SEND-TRAILER-MAP.
* EXEC CICS
   SEND MAP('LSTMAP3')
   MAPSET('LSTSET1')
   MAPONLY
   ACCUM
   PAGING
   ERASE
END-EXEC.

* 2230-SEND-HEADER-MAP.
* EXEC CICS
   SEND MAP('LSTMAP1')
   MAPSET('LSTSET1')
   FROM(LSTMAP1O)
   ACCUM
   PAGING
   ERASE
END-EXEC.

ADD 1 TO PAGE-NO.
The START command

Function

Interval control. The START command initiates another task that will begin execution when a specified time period has expired. Optionally, the START command can pass data to the task.

Syntax

```
EXEC CICS START TRANSID(name)
  [ { INTERVAL(hhmms) } ]
  [ TIME(hhmms) ]
  [ AFTER [HOURS(hh)] [MINUTES(mins)] [SECONDS(secs)] ]
  [ AT [HOURS(hh)] [MINUTES(mins)] [SECONDS(secs)] ] ]
  [ TERMDID(name) | USERID(data-value) ]
  [ SYSID(systemname) ]
  [ REQID(name) ]
  [ FROM(data-area)
    [ LENGTH(data-value) ] ]
  [ RTRANSID(name) ]
  [ RTERMDID(name) ]
  [ QUEUE(name) ]
  [ NOCHECK ]
  [ PROTECT ]
END-EXEC.
```

Options

*Note: If INTERVAL, TIME, AFTER, and AT are all omitted, INTERVAL(0) is assumed.*

TRANSID Specifies the 1- to 4-character transaction identifier that will be used to start the task.

INTERVAL Specifies a time interval; the task will be started when this interval has elapsed. You can code a literal in the form *hhmms*; leading zeros can be omitted. Or, you can code a data name for a 7-digit packed-decimal field (PIC S9(7) COMP-3); its value must be in the form *0hhmms*.

TIME Specifies a time of day when the task will be started. You can code a literal in the form *hhmms*; leading zeros can be omitted. Or, you can code a data name for a 7-digit packed-decimal field (PIC S9(7) COMP-3); its value must be in the form *0hhmms*.

AFTER Specifies that the HOURS, MINUTES, and SECONDS options indicate a duration after which the task will be started.

AT Specifies that the HOURS, MINUTES, and SECONDS options indicate a time of day when the task will be started.

HOURS Specifies a binary fullword (PIC S9(8) COMP) in the range of 0 to 99.
MINUTES Specifies a binary fullword (PIC S9(8) COMP) in the range of 0 to 59 or 0 to 5999.

SECONDS Specifies a binary fullword (PIC S9(8) COMP) in the range of 0 to 59 or 0 to 359999.

TERMID Specifies a 1- to 4-character terminal identifier that identifies the terminal where the started task will be attached. If specified, it must be defined in the Terminal Control Table (TCT). If omitted, the task is not attached to any terminal and, as a result, can’t do any terminal I/O.

USERID Specifies the user-id under whose authority the started task is to run if the task is not associated with a terminal. If TERMID is used instead, the user-id defaults to userid1.

SYSID Specifies the 1- to 4-character name of a remote system where the task is to be started.

REQID Specifies a 1- to 8-character name used to uniquely identify this START command. If specified, a CANCEL command can be issued later to cancel the task before it begins executing. (Once the task has started, however, the CANCEL command has no effect.)

FROM Specifies a data area that contains data to be passed to the started task. The started task receives this data by issuing a RETRIEVE command with the INTO or SET option.

LENGTH Specifies a binary halfword (PIC S9(4) COMP) or literal value that indicates the length of the FROM area.

RTRANSID Specifies a 1- to 4-character name that’s passed to the started task. The started task receives the name by issuing a RETRIEVE command with the RTRANSID option.

RTERMID Specifies a 1- to 4-character name that’s passed to the started task. The started task receives the name by issuing a RETRIEVE command with the RTERMID option.

QUEUE Specifies a 1- to 8-character name that’s passed to the started task. The started task receives the name by issuing a RETRIEVE command with the QUEUE option.

NOCHECK Specifies that when the started task is to be initiated on another system, the task issuing the START command should not wait for confirmation that the START command was successfully processed.

PROTECT Specifies that the task can not be started until the task issuing the START command issues a syncpoint, either by ending or by issuing a SYNCPOINT command.

### Exceptional conditions

*Note: The default action for these conditions is to terminate the task.*

**INVREQ** The START command is invalid, the specified hours, minutes, or seconds are out of range, or the REQID name already exists.

**IOERR** An I/O error occurred.

**ISCINVREQ** An undeterminable error occurred on the remote system specified in the SYSID option.
LENGERR A length error occurred.
NOTAUTH The current transaction’s PCT entry specified that resource security checking should be done, and the operator is not authorized to access the transaction to be started.
SYSIDERR The system identified by SYSID could not be located or accessed.
TERMIDERR The terminal identified by the TERMID option isn’t defined in the Terminal Control Table (TCT).
TRANSIDERR The transaction identified by the TRANSID option isn’t defined in the Program Control Table (PCT).
USERIDERR The user-id specified in the USERID option is not known to the external security manager.

Notes and tips

- The START command has three common uses:
  The first is when an application function is divided into two or more independent programs that can be executed simultaneously because they don’t depend on one another. In this case, use one or more START commands to immediately start one or more tasks.
  The second is when you need to begin a task at some time in the future. For example, you might use a START command to schedule a task for execution at 6:00 a.m., when the system’s usage is low. To do that, you can issue a START command with the TIME or AT option. Or you might have a program that needs to restart itself at regular intervals. In that case, the program can issue a START command with the INTERVAL or AFTER option before it ends.
  The third is to implement a menu structure. When you use the XCTL command to execute applications from a menu, the user and the application continue to run under the trans-id of the menu transaction. In contrast, you can use the START command if you want to run under the trans-id of the application program chosen from the menu instead.
- Usually, if two or more START commands specify the same expiration time and the same trans-id, one task will be started for each START command. However, if data is passed to the started task, only one task is started. If this task repeatedly issues RETRIEVE commands to process all of the data sent to it, then no additional tasks are started. But if it does not, the task is started again and again until all of the passed data has been processed.
- There are two ways to use the HOURS, MINUTES, and SECONDS options following AFTER. If you use them in combination, the ranges are 0 to 99 for HOURS, 0 to 59 for MINUTES, and 0 to 59 for SECONDS. However, if you specify only one option, you can use the larger ranges: 0 to 99 for HOURS, 0 to 5999 for MINUTES, and 0 to 359999 for SECONDS. For example, you could specify AFTER MINUTES(1) SECONDS(30), or you could specify AFTER SECONDS(90). Both have the same effect.
Coding example (no data, no terminal)

These examples show how to issue a START command to start the transaction named RFK4 at 6:30 a.m. The first uses the TIME option, the second uses the AT option. No data is passed to the task, and no terminal is associated with the task.

EXEC CICS
  START TRANSID('RFK4')
      TIME(063000)
  END-EXEC.

EXEC CICS
  START TRANSID('RFK4')
      AT HOURS(6) MINUTES(30)
  END-EXEC.

Coding example (data, terminal)

These examples show how to start a transaction named DKM3 in 10 minutes, passing it the 100 bytes of data in the field named DKM3-DATA. The first uses the INTERVAL option, the second uses the AFTER option. The task will be attached to the terminal named L580.

EXEC CICS
  START TRANSID('DKM3')
      INTERVAL(1000)
      TERMID('L580')
      FROM(DKM3-DATA)
      LENGTH(100)
  END-EXEC.

EXEC CICS
  START TRANSID('DKM3')
      AFTER MINUTES(10)
      TERMID('L580')
      FROM(DKM3-DATA)
      LENGTH(100)
  END-EXEC.
The WEB READ FORMFIELD command

Function

Web services. (TS 2.1 and later) The WEB READ command retrieves the value of a specified field from an HTML form. The name of the field to be extracted is given in the FORMFIELD option.

Syntax

```cicstex
EXEC CICS
WEB READ    FORMFIELD(data-area)
[ NAMELENGTH(data-value) ]
{ VALUE(data-area) | SET(pointer-ref) }
VALUELENGTH(data-area)
[ CLNTCODEPAGE (name)       
  HOSTCODEPAGE (name) ]
END-EXEC.
```

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMFIELD</td>
<td>Specifies the name of the form field to extract a value from. CICS will find the matching name on the HTML form and return the value associated with the field. The name you specify here is not case-sensitive.</td>
</tr>
<tr>
<td>NAMELENGTH</td>
<td>Specifies a binary fullword (PIC S9(8) COMP) or literal value that gives the length of the name in the FORMFIELD option.</td>
</tr>
<tr>
<td>VALUE</td>
<td>Specifies a data area that receives the form field value.</td>
</tr>
<tr>
<td>SET</td>
<td>Specifies a binary fullword (PIC S9(8) COMP) where the address of the received data is placed.</td>
</tr>
<tr>
<td>VALUELENGTH</td>
<td>Specifies a binary fullword (PIC S9(8) COMP) or literal value that gives the length of the form field value.</td>
</tr>
<tr>
<td>CLNTCODEPAGE</td>
<td>Specifies the 40-character name of the code page used when data is converted from the client code page. When you use this option, you must also specify the HOSTCODEPAGE option.</td>
</tr>
<tr>
<td>HOSTCODEPAGE</td>
<td>Specifies the 8-character name of the host code page that’s used when the forms data is converted from the ASCII code page. When you use this option, you must also specify the CLNTCODEPAGE option.</td>
</tr>
</tbody>
</table>
Exceptional conditions

*Note: The default action for these conditions is to terminate the task.*

INVREQ The program is a non-CICS Web interface application, no forms were provided in the body of the HTTP request, or the codepage combination for the client and server is invalid.

LENGERR The length on VALUELENGTH is less than or equal to zero.

NOTFND A form field with the specified name can’t be found.

Notes and tips

- IBM introduced the WEB commands in CICS TS 1.3 as a way of simplifying the communication between CICS and an external web environment. Prior to TS 1.3, the HTML template manager was required to process HTTP header and HTML form information.
- Provided that you know the names of the form fields in the HTML document, the WEB READ command allows you to process each field individually. If you don’t know the names of the form fields, or if you need to process all of the fields on the HTML form, use the WEB READNEXT FORMFIELD command in a browse operation instead.
- A host code page and a client code page are used to translate data to the format that’s used on the server and the client, respectively. In an IBM environment, usually the host uses EBCDIC code while the clients use ASCII. Specifying what the host and client code pages are if they’re not standard EBCDIC and ASCII will ensure that the information being transmitted will be translated correctly.

Coding example

The following example shows how you can read in the value associated with the form field named “BOOKNO” from the HTML form.

```
1000-WEB-READ.
*
MOVE 'BOOKNO' TO FORM-FIELD-NAME.
MOVE LENGTH OF FORM-FIELD-NAME TO FORM-FIELD-NAME-LEN.
MOVE LENGTH OF FORM-FIELD-VALUE TO FORM-FIELD-VALUE-LEN.
EXEC CICS
  WEB READ FORMFIELD (FORM-FIELD-NAME)
  NAMELENGTH (FORM-FIELD-NAME-LEN)
  VALUE (FORM-FIELD-VALUE)
  VALUELENGTH (FORM-FIELD-VALUE-LEN)
END-EXEC.
```