

CICS Transaction Server for z/OS



CICS Debugging Tools Interfaces Reference

Version 3 Release 1

CICS Transaction Server for z/OS



CICS Debugging Tools Interfaces Reference

Version 3 Release 1

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 29.

This edition applies to Version 3 Release 1 of CICS Transaction Server for z/OS, program number 5655-M15, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

© Copyright IBM Corporation 2002, 2010.

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

Contents

Preface	v
What this book is about.	v
Who this book is for	v
What you need to know to use this book	v
Notes on terminology	v
 Chapter 1. The debugging tools sockets interface.	 1
Overview of the debugging tools sockets interface	1
Setting up CICS to use the debugging tools sockets interface	1
Using the debugging tools sockets interface	1
Code page conversion	2
Environmental restrictions and programming requirements	2
CALL instruction programming interface	2
Assembler Language Call Format	2
Code CALL Instructions	2
ACCEPT	2
BIND	4
CLOSE	5
CONNECT	5
GETHOSTBYNAME	7
GETHOSTID	8
GETSOCKNAME	8
INITAPI	9
LISTEN	10
READ	11
SHUTDOWN	12
SOCKET	13
WRITE	14
Return codes	15
 Chapter 2. The debugging tools pattern matching interface	 19
Invoking the pattern matching interface	19
COMMAREA structure for pattern matching	19
 Bibliography	 21
The CICS Transaction Server for z/OS library	21
The entitlement set	21
PDF-only books	21
Other CICS books	23
Determining if a publication is current	23
 Accessibility	 25
 Index	 27
 Notices	 29
Trademarks	30
 Sending your comments to IBM	 31

Preface

What this book is about

This book describes the debugging tools interfaces for CICS® Transaction Server for z/OS®, Version 3 Release 1. The debugging tools interfaces are assembler language programming interfaces that allow debugging tools to use CICS functions that are not available in the application programming interface. The interfaces are:

- The debugging tools sockets interface
- The debugging tools pattern matching interface

Who this book is for

Assembler language programmers who are writing debugging tools that work with CICS application programs.

What you need to know to use this book

- You should have a good knowledge of Assembler Language programming in the CICS environment.
- To use the debugging tools sockets interface, you should be familiar with programming sockets programs for TCP/IP.
- To use the pattern matching interface, you should be familiar with the use of debugging profiles to select programs for debugging.

Notes on terminology

The following abbreviations are used throughout this book:

Term	Meaning
CICS	When used without qualification in the book, refers to the CICS element of CICS Transaction Server for z/OS

Chapter 1. The debugging tools sockets interface

Overview of the debugging tools sockets interface

The debugging tools sockets interface is an interface which debugging tools can use to communicate with a debugger client. It uses the support for TCP/IP provided by the CICS sockets domain.

The interface supports a limited number of socket calls used in a restricted way, and is not a full function application programming interface. The interface is not optimized for concurrent use.

Setting up CICS to use the debugging tools sockets interface

To use the debugging tools sockets interface:

- Specify TCPIP=YES in your system initialization parameters.

The debugging tools sockets interface does not use a TCPIP SERVICE definition; however, you must ensure that the port numbers that you use for the sockets interface are different from those that you define in your TCPIP SERVICES.

Using the debugging tools sockets interface

The debugging tools sockets interface supports the protocols between a TCP/IP client and a TCP/IP server shown in Figure 1.

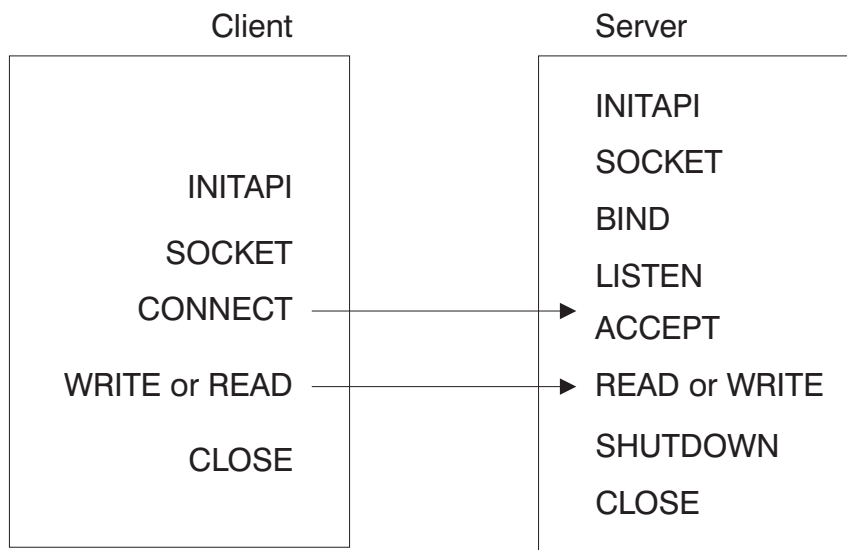


Figure 1. Protocols between client and server

In addition, the client and the server can issue the following calls:

GETHOSTID
GETHOSTBYNAME
GETSOCKNAME

The WRITE and READ calls can be repeated as often as required, and can be used to send data in either direction.

Code page conversion

The debugging tools sockets interface does not provide data conversion between ASCII and EBCDIC code pages. It is your responsibility to provide the necessary conversion between the EBCDIC code page use in your CICS system and the code page used in the debugging client.

Environmental restrictions and programming requirements

The following environmental restrictions and programming requirements apply to the debugging tools sockets interface:

- **SRB mode**
This interface may only be invoked in TCB mode (task mode).
- **Cross-memory mode**
This interface may only be invoked in a non-cross-memory environment (PASN=SASN=HASN).
- **Functional Recovery Routine (FRR)**
Do not invoke this interface with an FRR set. This will cause system recovery routines to be bypassed and severely damage the system.
- **Storage**
Storage acquired for the purpose of containing data returned from a socket call must be obtained in the same key as the program status word (PSW) at the time of the socket call.
- **Nested socket calls**
You can not issue nested socket calls within the same task. That is, if a request block (RB) issues a socket call and is interrupted by an interrupt request block (IRB) in an STIMER exit, any additional socket calls that the IRB attempts to issue are detected and flagged as an error.

CALL instruction programming interface

This section describes the general form of the CALL instruction for programs written in System/370 Assembler. The format and parameters are described for each socket call.

For more information about sockets, refer to the *UNIX Programmer's Reference Manual*.

The entry point for the CICS Sockets Extended module (DFHSOKET) is within the DFHSOCI module, which should be included explicitly in your link-editing JCL.

Assembler Language Call Format

Use the following 'DFHSOKET' call format for assembler language programs in order to meet the CICS requirement for quasi-reentrant programming:

►►—CALL DFHSOKET,(SOC_FUNCTION,—parm1, parm2, ...—ERRNO RETCODE),VL,MF=(E, PARMLIST)—►►

PARMLIST

A remote parameter list defined in dynamic storage DFHEISTG. This list contains addresses of the parameters that are referenced by the CALL.

Code CALL Instructions

This section contains the description, syntax, parameters, and other related information for each call instruction included in the debugging tools sockets interface.

ACCEPT

A server issues the ACCEPT call to accept a connection request from a client. The call points to a socket that was previously created with a SOCKET call and marked by a LISTEN call.

The ACCEPT call is a blocking call. When issued, the ACCEPT call:

1. Accepts the first connection on a queue of pending connections.
2. Creates a new socket with the same properties as an existing socket, and returns its descriptor in RETCODE. The original sockets remain available to the calling program to accept more connection requests.
3. The address of the client is returned in NAME for use by subsequent server calls.

Notes:

1. If the queue has no pending connection requests, ACCEPT blocks the socket.
2. The interface does not provide a function for screening clients. As a result, it is up to the program to control which connection requests it accepts, but it can close a connection immediately after discovering the identity of the client.

Figure 2 shows an example of ACCEPT call instructions.

```
SOC_FUNCTION DC CL16'ACCEPT'
S           DS H
NAME        DS 0XL16
FAMILY       DS H
PORT         DS H
IP_ADDRESS   DS F
RESERVED     DS CL8
ERRNO        DS F
RETCODE      DS F

CALL DFHSOKET, (SOC_FUNCTION,S,NAME,ERRNO,RETCODE)
```

Figure 2. ACCEPT Call Instructions Example

Input parameters

SOC_FUNCTION

A 16-byte character field containing 'ACCEPT'. Left-justify the field and pad it on the right with blanks.

S A halfword binary number specifying the descriptor of a socket that was previously created with a SOCKET call. In a concurrent server, this is the socket upon which the server listens.

Output parameters

NAME A socket address structure that contains the client's socket address.

FAMILY

A halfword binary field specifying the addressing family. The call returns the value 2 for AF_INET.

PORT A halfword binary field that is set to the client's port number.

IP_ADDRESS

A fullword binary field that is set to the 32-bit internet address, in network-byte-order, of the client's host machine.

RESERVED

Specifies eight bytes of binary zeros. This field is required, but not used.

ERRNO

A fullword binary field. If RETCODE is negative, the field contains an error number. See "Return codes" on page 15 for information about ERRNO return codes.

RETCODE

If the RETCODE value is positive, the RETCODE value is the new socket number.

If the RETCODE value is negative, check the ERRNO field for an error number.

BIND

In a typical server program, the BIND call follows a SOCKET call and completes the process of creating a new socket.

The BIND call can either specify the required port or let the system choose the port. A listener program should always bind to the same well-known port, so that clients know what socket address to use when attempting to connect.

The BIND call can specify the networks from which it is willing to accept connection requests. The program can fully specify the network interface by setting the ADDRESS field to the internet address of a network interface. Alternatively, the program can use a *wildcard* to specify that it wants to receive connection requests from any network interface. This is done by setting the ADDRESS field to a fullword of zeros.

Figure 3 shows an example of BIND call instructions.

```
SOC_FUNCTION DC CL16'BIND'
S           DS H
NAME        DS 0XL16
FAMILY      DS H
PORT        DS H
IP_ADDRESS  DS F
RESERVED    DS CL8
ERRNO       DS F
RETCODE     DS F

CALL DFHSOKET,(SOC_FUNCTION,S,NAME,ERRNO,RETCODE)
```

Figure 3. BIND Call Instruction Example

Input parameters

SOC_FUNCTION

A 16-byte character field containing BIND. The field is left-justified and padded to the right with blanks.

S A halfword binary number specifying the socket descriptor for the socket to be bound.

NAME Specifies the socket address structure for the socket that is to be bound.

FAMILY

A halfword binary field specifying the addressing family. The value is always set to 2, indicating AF_INET.

PORT A halfword binary field that is set to the port number to which you want the socket to be bound.

Note: If PORT is set to 0 when the call is issued, the system assigns the port number for the socket. The program can call the GETSOCKNAME call after the BIND call to discover the assigned port number.

IP_ADDRESS

A fullword binary field that is set to the 32-bit internet address (network byte order) of the socket to be bound.

RESERVED

Specifies an eight-byte character field that is required but not used.

Output parameters

ERRNO

A fullword binary field. If RETCODE is negative, this field contains an error number. See “Return codes” on page 15, for information about ERRNO return codes.

RETCODE

A fullword binary field that returns one of the following:

Value	Description
0	Successful call
-1	Check ERRNO for an error code

CLOSE

The CLOSE call performs the following functions:

- The CLOSE call shuts down a socket and frees all resources allocated to it. If the socket refers to an open TCP connection, the connection is closed.

After an unsuccessful socket call, a CLOSE should be issued and a new socket should be opened. An attempt to use the same socket with another call results in a nonzero return code.

Figure 4 shows an example of CLOSE call instructions.

```
SOC_FUNCTION DC CL16'CLOSE'
S           DS H
ERRNO       DS F
RETCODE     DS F
```

```
CALL DFHSOKET,(SOC_FUNCTION,S,ERRNO,RETCODE)
```

Figure 4. CLOSE Call Instruction Example

Output parameters

SOC_FUNCTION

A 16-byte field containing CLOSE. Left-justify the field and pad it on the right with blanks.

S A halfword binary field containing the descriptor of the socket to be closed.

Input parameters

ERRNO

A fullword binary field. If RETCODE is negative, this field contains an error number. See “Return codes” on page 15, for information about ERRNO return codes.

RETCODE

A fullword binary field that returns one of the following:

Value	Description
0	Successful call
-1	Check ERRNO for an error code

CONNECT

The CONNECT call is issued by a client to establish connection with a server. The call performs two tasks:

1. It completes the binding process if a BIND call has not been previously issued.
2. It attempts to make a connection to a remote socket. This connection is necessary before data can be transferred.

The call sequence issued by the client and server is:

1. The *server* issues BIND and LISTEN to create a passive open socket.
2. The *client* issues CONNECT to request the connection.
3. The *server* accepts the connection on the passive open socket, creating a new connected socket.

The CONNECT call blocks the calling program until the connection is established, or until an error is received. The completion cannot be checked by issuing a second CONNECT call.

Figure 5 shows an example of CONNECT call instructions.

```
SOC_FUNCTION DC CL16'CONNECT'
S           DS H
NAME        DS 0XL16
FAMILY       DS H
PORT         DS H
IP_ADDRESS   DS F
RESERVED     DS CL8
ERRNO        DS F
RETCODE      DS F
```

```
CALL DFHSOKET,(SOC_FUNCTION,S,,NAME,ERRNO,RETCODE)
```

Figure 5. CONNECT Call Instruction Example

Input parameters

SOC_FUNCTION

A 16-byte field containing CONNECT. Left-justify the field and pad it on the right with blanks.

S A halfword binary number specifying the socket descriptor of the socket that is to be used to establish a connection.

NAME A structure that contains the socket address of the target to which the local client socket is to be connected.

FAMILY

A halfword binary field specifying the addressing family. Specify a value of 2, denoting AF_INET.

PORT A halfword binary field that is set to the server's port number in network byte order. For example, if the port number is 5000 in decimal, it is stored as X'1388' in hex.

IP_ADDRESS

A fullword binary field that is set to the 32-bit internet address of the server's host machine in network byte order. For example, if the internet address is 129.4.5.12 in dotted decimal notation, it would be represented as '8104050C' in hex.

RESERVED

Specifies an 8-byte reserved field. This field is required, but is not used.

Output parameters

ERRNO

A fullword binary field. If RETCODE is negative, this field contains an error number. See "Return codes" on page 15 for information about ERRNO return codes.

RETCODE

A fullword binary field that returns one of the following:

Value	Description
0	Successful call
-1	Check ERRNO for an error code

GETHOSTBYNAME

The GETHOSTBYNAME call returns the alias name and the internet address of a host whose domain name is specified in the call. A given host can have multiple alias names and multiple host internet addresses.

The debugging tools sockets interface tries to resolve the host name through a name server.

Figure 6 shows an example of GETHOSTBYNAME call instructions.

```
SOC_FUNCTION DC    CL16'GETHOSTBYNAME'
NAMELEN      DS    F
NAME         DS    CL255
HOSTENT      DS    F
RETCODE      DS    F

CALL DFHSOKET,(SOC_FUNCTION,NAMELEN,NAME,HOSTENT,RETCODE)
```

Figure 6. GETHOSTBYNAME Call Instruction Example

Input parameters

SOC_FUNCTION

A 16-byte character field containing 'GETHOSTBYNAME'. The field is left-justified and padded on the right with blanks.

NAMELEN

A value set to the length of the host name.

NAME A character string, up to 255 characters, set to a host name. This call returns the address of the HOSTENT structure for this name.

Output parameters

HOSTENT

A fullword binary field that contains the address of the HOSTENT structure.

RETCODE

A fullword binary field that returns one of the following:

Value	Description
0	Successful call
-1	An error occurred

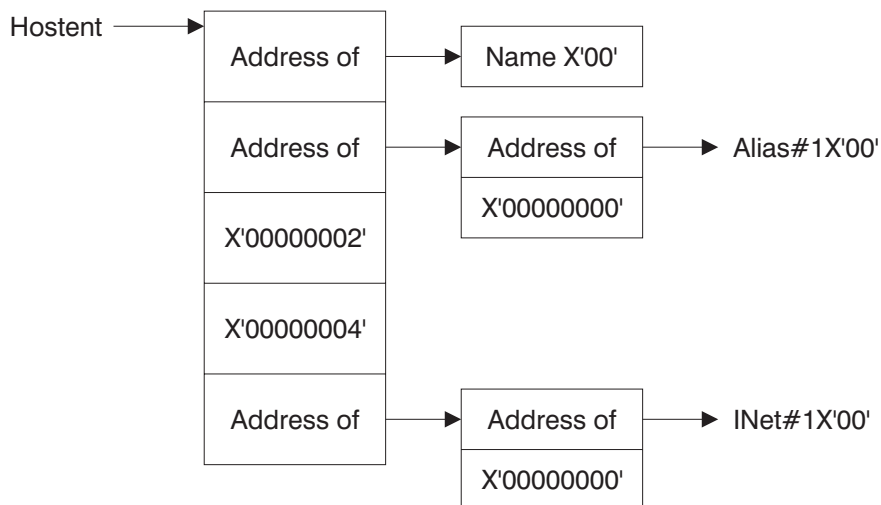


Figure 7. HOSTENT structure returned by the GETHOSTYBYNAME call

GETHOSTBYNAME returns the HOSTENT structure shown in Figure 7. This structure contains:

- The address of the host name that is returned by the call. The name length is variable and is ended by X'00'.
- The address of a list of addresses that point to the alias names returned by the call. This list is ended by the pointer X'00000000'. Each alias name is a variable length field ended by X'00'.
- The value returned in the FAMILY field is always 2 for AF_INET.
- The length of the host internet address returned in the HOSTADDR_LEN field is always 4 for AF_INET.
- The address of a list of addresses that point to the host internet addresses returned by the call. The list is ended by the pointer X'00000000'. If the call cannot be resolved, the HOSTENT structure contains the ERRNO 10214.

GETHOSTID

The GETHOSTID call returns the 32-bit internet address for the current host.

Figure 8 shows an example of GETHOSTID call instructions.

```

SOC_FUNCTION DC CL16'GETHOSTID'
RETCODE DS F

CALL DFHSOKET,(SOC_FUNCTION,RETCODE)
  
```

Figure 8. GETHOSTID Call Instruction Example

Output parameters

SOC_FUNCTION

A 16-byte character field containing 'GETHOSTID'. The field is left-justified and padded on the right with blanks.

RETCODE

Returns a fullword binary field containing the 32-bit internet address of the host. There is no ERRNO parameter for this call.

GETSOCKNAME

The GETSOCKNAME call returns the address currently bound to a specified socket. If the socket is not currently bound to an address, the call returns with the FAMILY field set, and the rest of the structure set to 0.

Since a socket is not assigned a name until after a successful call to either BIND, CONNECT, or ACCEPT, the GETSOCKNAME call can be used after an implicit bind to discover which port was assigned to the socket.

Figure 9 shows an example of GETSOCKNAME call instructions.

```
SOC_FUNCTION DC CL16'GETSOCKNAME'
S           DS H
NAME        DS 0XL16
FAMILY      DS H
PORT        DS H
IP_ADDRESS  DS F
RESERVED    DS CL8
ERRNO       DS F
RETCODE     DS F

CALL DFHSOKET,(SOC_FUNCTION,S,NAME,ERRNO,RETCODE)
```

Figure 9. GETSOCKNAME Call Instruction Example

Input parameters

SOC_FUNCTION

A 16-byte character field containing GETSOCKNAME. The field is left-justified and padded on the right with blanks.

S A halfword binary number set to the descriptor of a local socket whose address is required.

Output parameters

NAME Specifies the socket address structure returned by the call.

FAMILY

A halfword binary field containing the addressing family. The call always returns the value 2, indicating AF_INET.

PORT A halfword binary field set to the port number bound to this socket. If the socket is not bound, zero is returned.

IP_ADDRESS

A fullword binary field set to the 32-bit internet address of the local host machine.

RESERVED

Specifies eight bytes of binary zeros. This field is required but not used.

ERRNO

A fullword binary field. If RETCODE is negative, the field contains an error number. See “Return codes” on page 15 for information about ERRNO return codes.

RETCODE

A fullword binary field that returns one of the following:

Value	Description
0	Successful call
-1	Check ERRNO for an error code

INITAPI

The INITAPI call connects a program to the debugging tools sockets interface. All sockets programs must issue the INITAPI call before they issue other sockets calls.

Figure 10 on page 10 shows an example of INITAPI call instructions.

SOC_FUNCTION	DC	CL16'INITAPI'
MAXSOC	DS	H
IDENT	DS	0CL16
TCPNAME	DS	CL8
ADSNAME	DS	CL8
SUBTASK	DS	CL8
MAXSNO	DS	F
ERRNO	DS	F
RETCODE	DS	F

CALL DFHSOKET, (SOC_FUNCTION, MAXSOC, IDENT, SUBTASK, MAXSNO, ERRNO, RETCODE)

Figure 10. INITAPI Call Instruction Example

Input parameters

SOC_FUNCTION

A 16-byte character field containing INITAPI. The field is left-justified and padded on the right with blanks.

MAXSOC

A halfword binary field set to the maximum number of sockets this program will ever have open at one time. The maximum number is 2000 and the minimum number is 50. This value is used to determine the amount of memory that will be allocated for socket control blocks and buffers. If fewer than 50 sockets are requested, MAXSOC defaults to 50.

Note: This is not the same as the MAXSOCKETS system initialization parameter.

IDENT A structure containing the identities of the address space and the calling program's address space. Specify IDENT on the INITAPI call from an address space.

TCPNAME

Reserved — do not specify a value in this field.

ADSNAME

An 8-byte character field. Specify the name of the CICS startup job.

SUBTASK

Specify a null value (X'00000000') for this parameter.

Output parameters

MAXSNO

A fullword binary field that contains the highest socket number assigned to this program. The lowest socket number is zero. If you have 50 sockets, they are numbered from 0 to 49. If MAXSNO is not specified, the value for MAXSNO is 49.

ERRNO

A fullword binary field. If RETCODE is negative, the field contains an error number. See "Return codes" on page 15 for information about ERRNO return codes.

RETCODE

A fullword binary field that returns one of the following:

Value	Description
-------	-------------

0	Successful call
---	-----------------

-1	Check ERRNO for an error code
----	-------------------------------

LISTEN

The LISTEN call:

- Completes the bind, if BIND has not already been called for the socket.

- Creates a connection-request queue of a specified length for incoming connection requests.

Note: The LISTEN call is not supported for datagram sockets or raw sockets.

The LISTEN call is used by a server to receive connection requests from clients. When a connection request is received, a new socket is created by a subsequent ACCEPT call, and the original socket continues to listen for additional connection requests. The LISTEN call converts an active socket to a passive socket and conditions it to accept connection requests from clients. Once a socket becomes passive, it cannot initiate connection requests.

Figure 11 shows an example of LISTEN call instructions.

```
SOC_FUNCTION DC CL16'LISTEN'
S           DS H
BACKLOG     DS F
ERRNO       DS F
RETCODE     DS F

CALL DFHSOCKET, (SOC_FUNCTION,S,BACKLOG,ERRNO,RETCODE)
```

Figure 11. LISTEN Call Instruction Example

Input parameters

SOC_FUNCTION

A 16-byte character field containing LISTEN. The field is left-justified and padded to the right with blanks.

S A halfword binary number set to the socket descriptor.

BACKLOG

A fullword binary number set to the number of communication requests to be queued. Specify a value of 5 for this parameter.

Output parameters

ERRNO

A fullword binary field. If RETCODE is negative, the field contains an error number. See “Return codes” on page 15 for information about ERRNO return codes.

RETCODE

A fullword binary field that returns one of the following:

Value	Description
0	Successful call
-1	Check ERRNO for an error code

READ

The READ call reads the data on a socket.

Data is processed as streams of information with no boundaries separating the data. For example, if programs A and B are connected and program A sends 1000 bytes, each call to this function can return any number of bytes up to the entire 1000 bytes. The number of bytes returned will be contained in RETCODE. Therefore, programs should place this call in a loop that repeats until all data has been received.

Figure 12 on page 12 shows an example of READ call instructions.

```

SOC_FUNCTION DC CL16'READ'
S           DS H
NBYTE      DS F
BUF        DS CL(length of buffer).
ERRNO      DS F
RETCODE    DS F

CALL DFHSOCKET,(SOC_FUNCTION,S,NBYTE,BUF,ERRNO,RETCODE)

```

Figure 12. READ Call Instruction Example

Input parameters

SOC_FUNCTION

A 16-byte character field containing READ. The field is left-justified and padded to the right with blanks.

S A halfword binary number set to the socket descriptor of the socket that is going to read the data.

NBYTE

A fullword binary number set to the size of BUF. READ does not return more than the number of bytes of data in NBYTE even if more data is available.

Output parameters

BUF On input, a buffer to be filled by completion of the call. The length of BUF must be at least as long as the value of NBYTE.

ERRNO

A fullword binary field. If RETCODE is negative, the field contains an error number. See "Return codes" on page 15 for information about ERRNO return codes.

RETCODE

A fullword binary field that returns one of the following:

Value Description

- 0** A 0 return code indicates that the connection is closed and no data is available.
- >0** A positive value indicates the number of bytes copied into the buffer.
- 1** Check ERRNO for an error code.

SHUTDOWN

One way to terminate a network connection is to issue the CLOSE call which attempts to complete all outstanding data transmission requests prior to breaking the connection. The SHUTDOWN call can be used to close one-way traffic while completing data transfer in the other direction. The HOW parameter determines the direction of traffic to shutdown.

If you issue SHUTDOWN for a socket that currently has outstanding socket calls pending, see Table 1 to determine the effects of this operation on the outstanding socket calls.

Table 1. Effect of Shutdown Socket Call

Socket calls in local program	Local Program		Remote Program	
	Shutdown END_TO	Shutdown END_FROM	Shutdown END_FROM	Shutdown END_TO
Write calls	Error number EPIPE on first call		Error number EPIPE on second call*	
Read calls		Zero length return code		Zero length return code

Table 1. Effect of Shutdown Socket Call (continued)

* If you issue two write calls immediately, both might be successful, and an EPIPE error number might not be returned until a third write call is issued.

Figure 13 shows an example of SHUTDOWN call instructions.

```

SOC_FUNCTION DC CL16'SHUTDOWN'
S           DS H
HOW         DS F
END_FROM    EQU 0
END_TO      EQU 1
END_BOTH    EQU 2
ERRNO       DS F
RETCODE     DS F

CALL DFHSOCKET, (SOC_FUNCTION, S, HOW, ERRNO, RETCODE)

```

Figure 13. SHUTDOWN Call Instruction Example

Input parameters

SOC_FUNCTION

A 16-byte character field containing SHUTDOWN. The field is left-justified and padded on the right with blanks.

S A halfword binary number set to the socket descriptor of the socket to be shutdown.

HOW A fullword binary field. Set to specify whether all or part of a connection is to be shut down. The following values can be set:

Value	Description
-------	-------------

0 (END_FROM)

Ends further receive operations.

1 (END_TO)

Ends further send operations.

2 (END_BOTH)

Ends further send and receive operations.

Output parameters

ERRNO

A fullword binary field. If RETCODE is negative, the field contains an error number. See “Return codes” on page 15 for information about ERRNO return codes.

RETCODE

A fullword binary field that returns one of the following:

Value	Description
-------	-------------

0 Successful call

-1 Check ERRNO for an error code

SOCKET

The SOCKET call creates an endpoint for communication and returns a socket descriptor representing the endpoint.

Figure 14 on page 14 shows an example of SOCKET call instructions.

```

SOC_FUNCTION DC CL16'SOCKET'
AF           DC F'2'
SOCTYPE     DS F
STREAM      EQU 1
PROTO       DS F
ERRNO       DS F
RETCODE     DS F

CALL DFHSOKET,(SOC_FUNCTION,AF,SOCTYPE,PROTO,ERRNO,RETCODE)

```

Figure 14. SOCKET Call Instruction Example

Input parameters

SOC_FUNCTION

A 16-byte character field containing 'SOCKET'. The field is left-justified and padded on the right with blanks.

AF A fullword binary field set to the addressing family. Specify a value of 2, denoting AF_INET.

SOCTYPE

A fullword binary field set to the type of socket required. Specify a value of 1, denoting *stream sockets*. Stream sockets provide sequenced, two-way byte streams that are reliable and connection-oriented. They support a mechanism for out-of-band data.

PROTO

Reserved — do not specify a value in this field. The interface uses a protocol of TCP.

Output parameters

ERRNO

A fullword binary field. If RETCODE is negative, the field contains an error number. See “Return codes” on page 15 for information about ERRNO return codes.

RETCODE

A fullword binary field that returns one of the following:

Value	Description
≥0	Contains the new socket descriptor
-1	Check ERRNO for an error code

WRITE

The WRITE call writes data on a connected socket.

Sockets act like streams of information with no boundaries separating data. For example, if a program wishes to send 1000 bytes, each call to this function can send any number of bytes, up to the entire 1000 bytes. The number of bytes sent will be returned in RETCODE. Therefore, programs should place this call in a loop, calling this function until all data has been sent.

Figure 15 on page 15 shows an example of WRITE call instructions.

```

SOC_FUNCTION DC CL16'WRITE'
S           DS H
NBYTE      DS F
BUF        DS CL(length of buffer)
ERRNO      DS F
RETCODE    DS F

CALL DFHSOKET,(SOC_FUNCTION,S,NBYTE,BUF,ERRNO,RETCODE)

```

Figure 15. WRITE Call Instruction Example

Input parameters

SOC_FUNCTION

A 16-byte character field containing WRITE. The field is left-justified and padded on the right with blanks.

S A halfword binary field set to the socket descriptor.

NBYTE

A fullword binary field set to the number of bytes of data to be transmitted.

BUF Specifies the buffer containing the data to be transmitted.

Output parameters

ERRNO

A fullword binary field. If RETCODE is negative, the field contains an error number. See “Return codes” for information about ERRNO return codes.

RETCODE

A fullword binary field that returns one of the following:

Value Description

- ≥0** A successful call. A return code greater than zero indicates the number of bytes of data written.
- −1** Check ERRNO for an error code.

Return codes

Table 2. Sockets return codes

Error number	Error description	Programmer's response
30001	Unknown session token	Call your IBM® Software Support Center
30002	Insufficient storage	Retry the request when CICS is not short on storage
30003	I/O error	Retry the request. Data might not be available at this time.
30004	Connection closed	Determine why the partner system has closed the connection, and retry the request
30005	No socket available	Retry the request when more sockets are available
30006	Client error	Call your IBM Software Support Center
30007	Invalid option	Call your IBM Software Support Center

Table 2. Sockets return codes (continued)

Error number	Error description	Programmer's response
30008	Missing option	Call your IBM Software Support Center
30009	Not authorized	Call your IBM Software Support Center
30010	State error	Call your IBM Software Support Center
30011	Never associated	Call your IBM Software Support Center
30012	Notification unavailable	Call your IBM Software Support Center
30013	Already associated	Call your IBM Software Support Center
30014	TCP not active	Ensure TCP/IP is active in your CICS region
30015	Scheduled	Should not occur. Call your IBM Software Support Center
30016	No connection	Retry the request when the partner system can accept connections
30017	Connection refused	Retry the request when the partner system can accept connections
30018	Address in use	Retry the request when the partner system can accept connections
30019	Address not available	Retry the request when the partner system can accept connections
30020	Insufficient threads	Increase the number of threads for each OMVS process
30021	Notified	Should not occur. Call your IBM Software Support Center
30022	Not pending	Should not occur. Call your IBM Software Support Center
30023	Lock failure	Call your IBM Software Support Center
30024	Socket in use	Retry the request when the partner system can accept connections
30025	Timed out	Determine why the request timed out and retry the request
30026	Task canceled	Determine why the task was canceled, and retry the request
30027	CEEPIPI error	Call your IBM Software Support Center
30028	Listener attach failure	Call your IBM Software Support Center
30029	TCP/IP unavailable	Ensure TCP/IP is active in your CICS region
30030	TCP/IP already open	Should not occur. Call your IBM Software Support Center

Table 2. Sockets return codes (continued)

Error number	Error description	Programmer's response
30031	TCP/IP already closed	Should not occur. Call your IBM Software Support Center
30032	Unknown listen token	Call your IBM Software Support Center
30033	Unknown session token	Call your IBM Software Support Center
30034	Unknown client token	Call your IBM Software Support Center
30035	Unknown server address	Should not occur. Call your IBM Software Support Center
30036	Unknown client hostname	Should not occur. Call your IBM Software Support Center
30037	Unknown server hostname	Should not occur. Call your IBM Software Support Center
30038	Hostname truncated	Should not occur. Call your IBM Software Support Center
30039	Repository error	Should not occur. Call your IBM Software Support Center
30040	MAXSOCKETS hard limit	Retry the request when more sockets are available
30041	At MAXSOCKETS	Retry the request when more sockets are available
30042	Unknown socket token	Call your IBM Software Support Center
30043	I/O error	Retry the request. Data might not be available at this time.
30044	IIOP listener no	Should not occur. Call your IBM Software Support Center
30045	INITAPI getmain array fail	CICS internal error. Call your IBM Software Support Center
30046	HOSTENT getmain fail	CICS internal error. Call your IBM Software Support Center
30047	SOCKNAME getmain fail	CICS internal error. Call your IBM Software Support Center
30048	Alias struct getmain fail	CICS internal error. Call your IBM Software Support Center
30049	Inet struct getmain fail	CICS internal error. Call your IBM Software Support Center
30050	Alias getmain fail	CICS internal error. Call your IBM Software Support Center
30051	Inet getmain fail	CICS internal error. Call your IBM Software Support Center
30052	No room in sock array	Increase the value of the MAXSOC parameter on the INITAPI request

Chapter 2. The debugging tools pattern matching interface

Use the debugging tools pattern matching interface to determine if a program instance that you specify matches an active debugging profile. The interface returns information about the profile that is the best match for the program instance you specify.

Invoking the pattern matching interface

To invoke the pattern matching interface, perform the following steps:

1. LINK to program DFHDPDC, with a commarea that has the structure described in the following section. The commarea must have a length of 699 bytes or longer.

COMMAREA structure for pattern matching

Offset Hex	Offset (Decimal)	Type	Length	Name	Type of data	Description
X'00'	0		16		Reserved	
X'10'	16		1		Input	Specify a value of X'02'
X'11'	17		1		Reserved	
X'12'	18	UNSIGNED	1	DPCC_RESPONSE	Output	X'01' The specified program instance matches an active debugging profile. X'02' The specified program instance does not match an active debugging profile.
X'13'	19	CHARACTER	4	DPCC_TRANID	Input	Specify the transaction ID that is used to identify matching profiles
X'17'	23	CHARACTER	4	DPCC_TERMID	Input	Specify the terminal ID that is used to identify matching profiles
X'1B'	27	CHARACTER	8	DPCC_PROGID	Input	Specify the program name that is used to identify matching profiles
X'23'	35	CHARACTER	30	DPCC_COMP_UNIT	Input	Specify the name of the compile unit that is used to identify matching profiles
X'41'	65	CHARACTER	8	DPCC_USERID	Input	Specify the user ID that is used to identify matching profiles
X'49'	73	CHARACTER	8	DPCC_NETNAME	Input	Specify the terminal Netname that is used to identify matching profiles
X'51'	81	CHARACTER	8	DPCC_APPLID	Input	Specify the APPLID that is used to identify matching profiles
X'59'	89	CHARACTER	1	DPCC_SESSION_TYPE	Output	X'01' The best matching debugging profile specifies a session type of 3270 X'02' The best matching debugging profile specifies a session type of TCP
X'5A'	90	CHARACTER	255	DPCC_IP_NAME_OR_ADDR	Output	For a session type of TCP, returns the TCP/IP name or address specified in the best matching profile
X'159'	345	CHARACTER	5	DPCC_PORT	Output	For a session type of TCP, returns the port number specified in the best matching profile

Offset Hex	Offset (Decimal)	Type	Length	Name	Type of data	Description
X'15E'	350	CHARACTER	4	DPCC_3270_DISPLAY	Output	For a session type of 3270, returns the terminal Id of the 3270 terminal specified in the best matching profile
X'162'	354	UNSIGNED	1	DPCC_TEST_LEVEL	Output	If the best matching profile is for a Language Environment® program, returns the Test Level specified in the profile
X'163'	355	CHARACTER	44	DPCC_COMMAND_FILE	Output	If the best matching profile is for a Language Environment program, returns the name of the Command File specified in the profile
X'18F'	399	UNSIGNED	1	DPCC_PROMPT	Output	If the best matching profile is for a Language Environment program, returns the Prompt Level specified in the profile
X'190'	400	CHARACTER	44	DPCC_PREFERENCE_FILE	Output	If the best matching profile is for a Language Environment program, returns the name of the Preference File specified in the profile
X'1BC'	444	CHARACTER	255	DPCC_LE_OPTIONS	Output	If the best matching profile is for a Language Environment program, returns the Language Environment options specified in the profile

Bibliography

The CICS Transaction Server for z/OS library

The published information for CICS Transaction Server for z/OS is delivered in the following forms:

The CICS Transaction Server for z/OS Information Center

The CICS Transaction Server for z/OS Information Center is the primary source of user information for CICS Transaction Server. The Information Center contains:

- Information for CICS Transaction Server in HTML format.
- Licensed and unlicensed CICS Transaction Server books provided as Adobe Portable Document Format (PDF) files. You can use these files to print hardcopy of the books. For more information, see “PDF-only books.”
- Information for related products in HTML format and PDF files.

One copy of the CICS Information Center, on a CD-ROM, is provided automatically with the product. Further copies can be ordered, at no additional charge, by specifying the Information Center feature number, 7014.

Licensed documentation is available only to licensees of the product. A version of the Information Center that contains only unlicensed information is available through the publications ordering system, order number SK3T-6945.

Entitlement hardcopy books

The following essential publications, in hardcopy form, are provided automatically with the product. For more information, see “The entitlement set.”

The entitlement set

The entitlement set comprises the following hardcopy books, which are provided automatically when you order CICS Transaction Server for z/OS, Version 3 Release 1:

Memo to Licensees, GI10-2559
CICS Transaction Server for z/OS Program Directory, GI10-2586
CICS Transaction Server for z/OS Release Guide, GC34-6421
CICS Transaction Server for z/OS Installation Guide, GC34-6426
CICS Transaction Server for z/OS Licensed Program Specification, GC34-6608

You can order further copies of the following books in the entitlement set, using the order number quoted above:

CICS Transaction Server for z/OS Release Guide
CICS Transaction Server for z/OS Installation Guide
CICS Transaction Server for z/OS Licensed Program Specification

PDF-only books

The following books are available in the CICS Information Center as Adobe Portable Document Format (PDF) files:

CICS books for CICS Transaction Server for z/OS

General

CICS Transaction Server for z/OS Program Directory, GI10-2586
CICS Transaction Server for z/OS Release Guide, GC34-6421
CICS Transaction Server for z/OS Migration from CICS TS Version 2.3, GC34-6425
CICS Transaction Server for z/OS Migration from CICS TS Version 1.3, GC34-6423
CICS Transaction Server for z/OS Migration from CICS TS Version 2.2, GC34-6424
CICS Transaction Server for z/OS Installation Guide, GC34-6426

Administration

CICS System Definition Guide, SC34-6428

CICS Customization Guide, SC34-6429
CICS Resource Definition Guide, SC34-6430
CICS Operations and Utilities Guide, SC34-6431
CICS Supplied Transactions, SC34-6432

Programming

CICS Application Programming Guide, SC34-6433
CICS Application Programming Reference, SC34-6434
CICS System Programming Reference, SC34-6435
CICS Front End Programming Interface User's Guide, SC34-6436
CICS C++ OO Class Libraries, SC34-6437
CICS Distributed Transaction Programming Guide, SC34-6438
CICS Business Transaction Services, SC34-6439
Java Applications in CICS, SC34-6440
JCICS Class Reference, SC34-6001

Diagnosis

CICS Problem Determination Guide, SC34-6441
CICS Messages and Codes, GC34-6442
CICS Diagnosis Reference, GC34-6899
CICS Data Areas, GC34-6902
CICS Trace Entries, SC34-6443
CICS Supplementary Data Areas, GC34-6905

Communication

CICS Intercommunication Guide, SC34-6448
CICS External Interfaces Guide, SC34-6449
CICS Internet Guide, SC34-6450

Special topics

CICS Recovery and Restart Guide, SC34-6451
CICS Performance Guide, SC34-6452
CICS IMS Database Control Guide, SC34-6453
CICS RACF Security Guide, SC34-6454
CICS Shared Data Tables Guide, SC34-6455
CICS DB2 Guide, SC34-6457
CICS Debugging Tools Interfaces Reference, GC34-6908

CICSplex SM books for CICS Transaction Server for z/OS

General

CICSplex SM Concepts and Planning, SC34-6459
CICSplex SM User Interface Guide, SC34-6460
CICSplex SM Web User Interface Guide, SC34-6461

Administration and Management

CICSplex SM Administration, SC34-6462
CICSplex SM Operations Views Reference, SC34-6463
CICSplex SM Monitor Views Reference, SC34-6464
CICSplex SM Managing Workloads, SC34-6465
CICSplex SM Managing Resource Usage, SC34-6466
CICSplex SM Managing Business Applications, SC34-6467

Programming

CICSplex SM Application Programming Guide, SC34-6468
CICSplex SM Application Programming Reference, SC34-6469

Diagnosis

CICSplex SM Resource Tables Reference, SC34-6470
CICSplex SM Messages and Codes, GC34-6471
CICSplex SM Problem Determination, GC34-6472

CICS family books

Communication

CICS Family: Interproduct Communication, SC34-6473

CICS Family: Communicating from CICS on System/390, SC34-6474

Licensed publications

The following licensed publications are not included in the unlicensed version of the Information Center:

CICS Diagnosis Reference, GC34-6899

CICS Data Areas, GC34-6902

CICS Supplementary Data Areas, GC34-6905

CICS Debugging Tools Interfaces Reference, GC34-6908

Other CICS books

The following publications contain further information about CICS, but are not provided as part of CICS Transaction Server for z/OS, Version 3 Release 1.

<i>Designing and Programming CICS Applications</i>	SR23-9692
<i>CICS Application Migration Aid Guide</i>	SC33-0768
<i>CICS Family: API Structure</i>	SC33-1007
<i>CICS Family: Client/Server Programming</i>	SC33-1435
<i>CICS Transaction Gateway for z/OS Administration</i>	SC34-5528
<i>CICS Family: General Information</i>	GC33-0155
<i>CICS 4.1 Sample Applications Guide</i>	SC33-1173
<i>CICS/ESA 3.3 XRF Guide</i>	SC33-0661

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both hardcopy and BookManager® softcopy versions of a publication are usually in step. However, due to the time required to print and distribute hardcopy books, the BookManager version is more likely to have had last-minute changes made to it before publication.

Subsequent updates will probably be available in softcopy before they are available in hardcopy. This means that at any time from the availability of a release, softcopy versions should be regarded as the most up-to-date.

For CICS Transaction Server books, these softcopy updates appear regularly on the *Transaction Processing and Data Collection Kit* CD-ROM, SK2T-0730-xx. Each reissue of the collection kit is indicated by an updated order number suffix (the -xx part). For example, collection kit SK2T-0730-06 is more up-to-date than SK2T-0730-05. The collection kit is also clearly dated on the cover.

Updates to the softcopy are clearly marked by revision codes (usually a # character) to the left of the changes.

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully.

You can perform most tasks required to set up, run, and maintain your CICS system in one of these ways:

- using a 3270 emulator logged on to CICS
- using a 3270 emulator logged on to TSO
- using a 3270 emulator as an MVS™ system console

IBM Personal Communications provides 3270 emulation with accessibility features for people with disabilities. You can use this product to provide the accessibility features you need in your CICS system.

Index

A

ACCEPT call
 on debugging tools sockets interface 2
AF parameter on call interface, on SOCKET 14

B

BACKLOG parameter on call interface, LISTEN call 11
BIND call
 on debugging tools sockets interface 4
BUF parameter on call socket interface 2

C

Call Instructions for Assembler, PL/1, and COBOL
 Programs 2
CLIENT parameter on call socket interface 2
CLOSE call
 on debugging tools sockets interface 5
COMMAND parameter on call socket interface 2
CONNECT call
 on debugging tools sockets interface 5

D

data translation, socket interface 2
debugging tools sockets interface
 ACCEPT call 2
 BIND call 4
 CLOSE call 5
 CONNECT call 5
 GETHOSTBYNAME call 7
 GETHOSTID call 8
 GETSOCKNAME call 8
 INITAPI call 9
 LISTEN call 10
 READ call 11
 SHUTDOWN call 12
 SOCKET call 13
 WRITE call 14

E

ERRNO parameter on call socket interface 2

F

FLAGS parameter on call socket interface 2

G

GETHOSTBYNAME call
 on debugging tools sockets interface 7
GETHOSTID call
 on debugging tools sockets interface 8

GETSOCKNAME call
 on debugging tools sockets interface 8

I

INITAPI call
 on debugging tools sockets interface 9
IOV parameter on call socket interface 2
IOVCNT parameter on call socket interface 2

L

LENGTH parameter on call socket interface 2
LISTEN call
 on debugging tools sockets interface 10

M

MAXSOC parameter on call socket interface 2
MSG parameter on call socket interface 2

N

NAME 3
NBYTE parameter on call socket interface 2

O

OPTNAME parameter on call socket interface 2
OPTVAL parameter on call socket interface 2

P

PROTO parameter on call interface, on SOCKET 14

R

READ call
 on debugging tools sockets interface 11
REQARG and RETARG parameter on call socket
 interface 2
RETCODE parameter on call socket interface 2

S

SHUTDOWN call
 on debugging tools sockets interface 12
SOCKET call
 on debugging tools sockets interface 13
sockets interface
 ACCEPT call 2
 BIND call 4
 CLOSE call 5
 CONNECT call 5
 GETHOSTBYNAME call 7
 GETHOSTID call 8

sockets interface (*continued*)

GETSOCKNAME call 8

INITAPI call 9

LISTEN call 10

READ call 11

SHUTDOWN call 12

SOCKET call 13

WRITE call 14

SOCTYPE parameter on call interface, on

SOCKET 14

T

TIMEOUT parameter on call socket interface 2

U

utility programs 2

W

WRITE call

on debugging tools sockets interface 14

Notices

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

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

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

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

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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

This publication 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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. 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.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and service names might be trademarks of IBM or other companies.

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To ask questions, make comments about the functions of IBM products or systems, or to request additional publications, contact your IBM representative or your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:
IBM United Kingdom Limited
User Technologies Department (MP095)
Hursley Park
Winchester
Hampshire
SO21 2JN
United Kingdom
- By fax:
 - From outside the U.K., after your international access code use 44–1962–816151
 - From within the U.K., use 01962–816151
- Electronically, use the appropriate network ID:
 - IBMLink: HURSLEY(IDRCF)
 - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.



Product Number: 5655-M15

GC34-6908-01



Spine information:



CICS TS for z/OS

CICS Debugging Tools Interfaces Reference

Version 3
Release 1