CICS Transaction Server for z/OS
Version 4 Release 1

Debugging Tools Interfaces Reference

GC34-7039-01

CICS Transaction Server for z/OS
Version 4 Release 1

Debugging Tools Interfaces Reference

GC34-7039-01

Note
Before using this information and the product it supports, read the information in "Notices" on page 25.

This edition applies to Version 4 Release 1 of CICS Transaction Server for z/OS (product number 5655-S97) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Preface

What this manual is about

This manual documents intended Programming Interfaces that allow the customer to write programs to obtain the services of Version 4 Release 1.

This manual describes the debugging tools interfaces for CICS® Transaction Server for z/OS®, Version 4 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 manual is for

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

What you need to know to use this manual

- 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 manual:

Term	Meaning
CICS	When used without qualification in the manual, refers to the CICS element of CICS Transaction Server for z/OS, Version 4 Release 1

Changes in CICS Transaction Server for z/OS, Version 4 Release 1

For information about changes that have been made in this release, please refer to *What's New* in the information center, or the following publications:

- *CICS Transaction Server for z/OS What's New*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.2*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.1*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 2.3*

Chapter 1. The debugging tools sockets interface

The debugging tools sockets interface is an interface that 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.

The client set of functions that are explained in the interface support both IPv4 and IPv6 addressing; however, the server set of functions support IPv4 addressing only.

Setting up CICS to use the debugging tools sockets interface

To use the debugging tools sockets interface, you must set a system initialization parameter.

About this task

To use the debugging tools sockets interface:

- Specify TCPIP=YES in your system initialization parameters.

The debugging tools sockets interface does not use a TCPIPSERVICE 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 TCPIPSERVICEs.

Using the debugging tools sockets interface

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

About this task

The protocols are shown in Figure 1 on page 2.

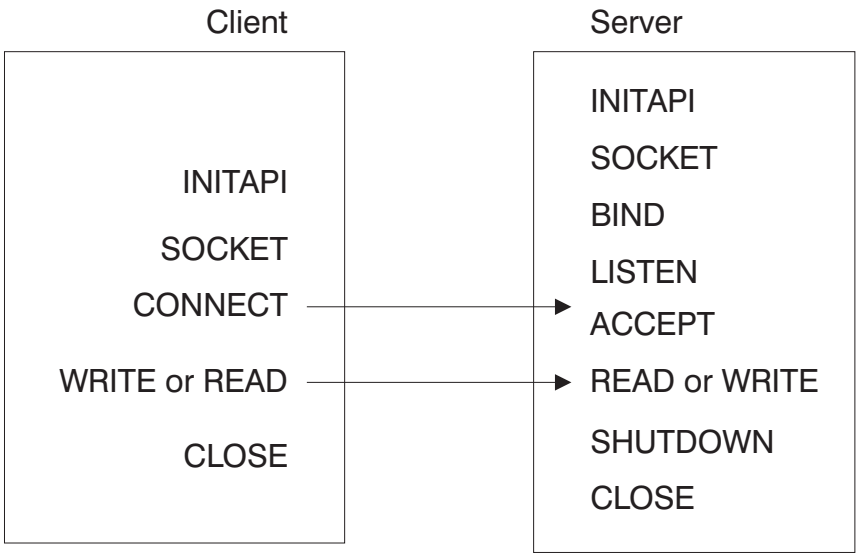

Figure 1. Protocols between client and server

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

Environmental restrictions and programming requirements apply to the debugging tools sockets interface.

SRB mode
The interface can only be invoked in TCB mode (task mode).

Cross-memory mode
The interface can only be invoked in a non-cross-memory environment (`PASN=SASN=HASN`).

Functional Recovery Routine (FRR)
The interface cannot be invoked this interface with an FRR set. Doing so 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

These topics describe the general form of the CALL instruction for programs written in 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

These topics contain 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 performs these functions:

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.

Note:

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

Example of ACCEPT call

```
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)
```

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, the server listens on this socket.

Output parameters

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

FAMILY
A halfword binary field specifying the addressing family. The call returns 2 for the AF_INET socket. For more information on AF_INET and AF_INET6, see the *z/OS® 1.9 Communications Server IPv6 Network and Application Design Guide*.

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

IP_ADDRESS
A fullword binary field that is set to the 32-bit IPv4 address, in network byte order, of the client host machine. IPv6 addressing is not supported.

RESERVED
Specifies 8 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 19 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 always binds to the same well-known port, so that clients know which socket address to use when attempting to connect.

The BIND call can specify the networks from which it will 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 will receive connection requests from any network interface. Set the ADDRESS field to a fullword of zeros for a wildcard.

Example of BIND call

```
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)
```

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 call returns 2 for the AF_INET socket. For more information on AF_INET and AF_INET6, see the *z/OS® 1.9 Communications Server IPv6 Network and Application Design Guide*.

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 IPv4 address (network byte order) of the socket to be bound. IPv6 addressing is not supported.

RESERVED
Specifies an 8-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 19, 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 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 unsuccesful 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.

Example of CLOSE call

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


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

Input 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.

Output parameters

ERRNO
A fullword binary field. If RETCODE is negative, this field contains an error number. See "Return codes" on page 19, 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 the following two tasks:

1. Completes the binding process if a BIND call has not been previously issued.

2. Attempts to make a connection to a remote socket. This connection is necessary before data can be transferred.

The following call sequence is issued by the client and server:

1. The *server* issues BIND and LISTEN calls to create a passive open socket.
2. The *client* issues a CONNECT call 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.

Example of CONNECT call

```
SOC_FUNCTION DC  CL16'CONNECT'
S            DS  H
NAME         DS  0XL28
FAMILY       DS  H
PORT         DS  H
IP_ADDRESS   DS  CL16
RESERVED     DS  CL8
ERRNO        DS  F
RETCODE      DS  F


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

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. FAMILY must match the value assigned to the AF field used in the SOCKET function request.

PORT A halfword binary field that is set to the server 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 16-byte field that is set to the IPv4 or IPv6 internet address of the socket to be bound. If FAMILY is set to 2 (denoting an AF_NET socket), the address is an IPv4 address and the first 4 bytes of IP_ADDRESS are used. For more information on AF_INET and AF_INET6, see the *z/OS® 1.9 Communications Server IPv6 Network and Application Design Guide*.

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 19 for information about ERRNO return codes.

RETCODE
A fullword binary field that returns one of the following values:

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

FREEADDRINFO

The FREEADDRINFO call frees the storage that was acquired by the z/OS Communications Server when the GETADDRINFO call was issued.

Example of FREEADDRINFO call

```
SOC_FUNCTION DC   CL16'FREEADDRINFO'
RESULTS      DS   A
ERRNO        DS   F
RETCODE      DS   F

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

Input parameters

SOC_FUNCTION
A 16-byte character field containing `FREEADDRINFO`. Left-justify the field and pad it on the right with blanks.

RESULTS
The name of a fullword field that contains a pointer to an **Addr_Info** structure or a linked list of **Addr_Info** structures returned by the GETADDRINFO command issued by the z/OS Communications Server.

Output parameters

ERRNO
A fullword binary field. If **RETCODE** is negative, the field contains an error number. For a list of return code values for FREEADDRINFO, see *z/OS Communications Server: IP and SNA Codes*.

RETCODE
A fullword binary field that returns one of the following values:

Value	Description
0	Successful call
-1	An error occurred.

GETADDRINFO

The GETADDRINFO call returns the 32-bit internet address for the current host from the GETADDRINFO command that is issued by z/OS Communications Server to resolve host or service name information. This command translates the name of a service location (host name) or a service name.

Example of GETADDRINFO call

```
SOC_FUNCTION DC   CL16'GETADDRINFO'
NAME         DS   CL255
NAMELEN      DS   F
SERVICE      DS   CL32
SERVICELEN   DS   F

HINTS        DS   A
RESULTS      DS   A
CANONICALLEN DS   F
ERRNO        DS   F
RETCODE      DS   F

             CALL DFHSOKET,(SOC_FUNCTION,NAME,NAMELEN,SERVICE,SERVICELEN,HINTS,RESULTS,
 CANONICALLEN,ERRNO,RETCODE)
```

Input parameters

SOC_FUNCTION

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

NAME

NAME is returned as one of the following strings:

- An EBCDIC character string, up to 255 characters long, set to the node name (host name) that is being queried.
- An EBCDIC character string set to the IP address of the node (host) where the service resides.

NAMELEN

The name of a fullword that contains the length of the NAME parameter.

SERVICE

SERVICE is returned as one of the following strings:

- An EBCDIC character string, up to 32 characters long, set to the service name that is being queried.
- An EBCDIC character string set to the port number of the required service.

SERVICELEN

The name of a fullword that contains the length of the SERVICE parameter.

HINTS

The name of a field that contains a pointer to a z/OS Communications Server input **Addr_Info** structure. The following fields can be specified in the **Addr_Info** structure:

- A set of flags (**ai_flags**) for interpreting the request. Here are the flags:
  - AI_PASSIVE
  - AI_CANONNAMEOK
  - AI_NUMERICHOST
  - AI_NUMERICSERV
  - AI_V4MAPPED
  - AI_ALL
  - AI_ADDRCONFIG

  For more information about **ai_flags**, see the Parameters topic in the *z/OS Communications Server IP CICS Sockets Guide*.

- The address family (**ai_family**) that the caller expects to be returned by the resolver. Here are the address families:
  - AF_UNSPEC
  - AF_INET
  - AF_INET6
- The socket type (**ai_socktype**) that the caller can accept as a response.
- The protocol (**ai_protocol**) that the caller can accept as a response.

All other fields in the **Addr_Info** structure must be set to zero.

If the **HINTS** parameter is not specified; that is, **HINT** is set to zero, the following settings are used:

- All flags are set to off.
- Address family is set to AF_UNSPEC.
- Socket type is set to 0.
- Protocol is set to 0.

Output parameters

RESULTS

The name of a field that contains a pointer to an output **Addr_Info** structure. If more than one address is returned, this field contains a linked list of output **Addr_Info** structures. Each output **Addr_Info** structure contains the following information about the information returned in the **Addr_Info** structure:

- A set of flags (**ai_flags**) for interpreting the address.
- The address family (**ai_family**) for the address.
- The socket type (**ai_socktype**) for the address.
- The protocol (**ai_protocol**) for the address.
- The length (**ai_addrlen**) of the **sock_inet_sockaddr** or **sock_inet6_sockaddr** structure returned in the **ai_addr** field.
- The canonical name (**ai_canonname**) associated with the **NAME** input parameter, if **NAME** was requested using the input AI_CANONNAMEOK flag. If more than one **Addr_Info** structure is returned, the canonical name is supplied in the first **Addr_Info** structure only.

CANONICALLEN

The name of a fullword binary field that contains the length of the canonical name that was returned in the first **Addr_Info** structure pointed to by the **RESULTS** parameter.

ERRNO

A fullword binary field. If **RETCODE** is negative, **ERRNO** contains an error number. For a list of return code values for GETADDRINFO, see *z/OS Communications Server: IP and SNA Codes*.

RETCODE

A fullword binary field that returns one of the following values:

Value	Description
0	Successful call
-1	An error occurred.

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.

Example of GETHOSTBYNAME call

```
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)
```

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

The HOSTENT structure

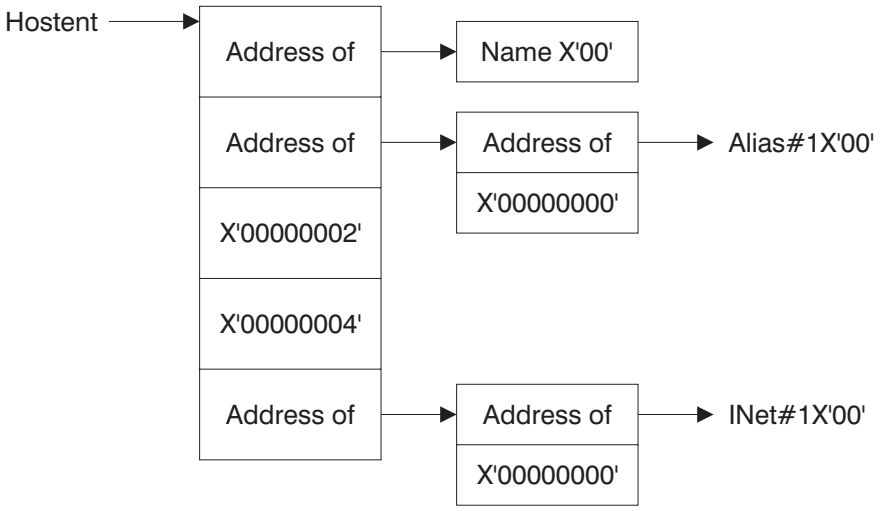

Figure 2. HOSTENT structure returned by the GETHOSTYBYNAME call

GETHOSTBYNAME returns the HOSTENT structure shown in Figure 2. 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.

Example of GETHOSTID call

```
SOC_FUNCTION DC   CL16'GETHOSTID'
RETCODE      DS   F

             CALL DFHSOKET,(SOC_FUNCTION,RETCODE)
```

Input parameters

SOC_FUNCTION

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

Output parameters

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.

Example of GETSOCKNAME call

```
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)
```

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 19 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.

Example of INITAPI call

```
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)
```

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 19 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, and creates a connection-request queue of a specified length for incoming connection requests.

Restriction: 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.

Example of LISTEN call

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

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

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 19 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.

Example of READ call

```
SOC_FUNCTION DC  CL16'READ'
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)
```

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 19 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 before 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
* 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.				

Example of SHUTDOWN call

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

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 19 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.

Example of SOCKET call

```
SOC_FUNCTION DC   CL16'SOCKET'
AF           DC   F'19'
SOCTYPE      DS   F
STREAM       EQU  1
PROTO        DS   F
ERRNO        DS   F
RETCODE      DS   F

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

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 19, denoting an AF_INET6 socket. You can specify a value of 2 for migration purposes however, the socket will be limited to IPv4 connections only. A halfword binary field specifying the addressing family. For more information on AF_INET and AF_INET6, see the *z/OS® 1.9 Communications Server IPv6 Network and Application Design Guide*.

SOCTYPE
A fullword binary field set to the type of socket required. Specify 1, denoting *stream sockets*. Stream sockets provide sequenced, 2-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 19 for information about ERRNO return codes.

RETCODE
A fullword binary field that returns one of the following values:

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 wants 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.

Example of WRITE call

```
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)
```

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

A table of the error numbers, error descriptions, and the suggested programmer's response.

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.

Error number	Error description	Programmer's response
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
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

Error number	Error description	Programmer's response
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
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

Error number	Error description	Programmer's response
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, LINK to program DFHDPCP, with a commarea.

Procedure

Use a commarea with a length of 699 bytes or longer and the following structure:

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

Offset Hex	Offset (Decimal)	Type	Length	Name	Type of data	Description
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
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

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 want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM 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. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

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

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Bibliography

CICS books for CICS Transaction Server for z/OS

General

CICS Transaction Server for z/OS Program Directory, GI13-0536
CICS Transaction Server for z/OS What's New, GC34-6994
CICS Transaction Server for z/OS Upgrading from CICS TS Version 2.3, GC34-6996
CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.1, GC34-6997
CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.2, GC34-6998
CICS Transaction Server for z/OS Installation Guide, GC34-6995

Access to CICS

CICS Internet Guide, SC34-7021
CICS Web Services Guide, SC34-7020

Administration

CICS System Definition Guide, SC34-6999
CICS Customization Guide, SC34-7001
CICS Resource Definition Guide, SC34-7000
CICS Operations and Utilities Guide, SC34-7002
CICS RACF Security Guide, SC34-7003
CICS Supplied Transactions, SC34-7004

Programming

CICS Application Programming Guide, SC34-7022
CICS Application Programming Reference, SC34-7023
CICS System Programming Reference, SC34-7024
CICS Front End Programming Interface User's Guide, SC34-7027
CICS C++ OO Class Libraries, SC34-7026
CICS Distributed Transaction Programming Guide, SC34-7028
CICS Business Transaction Services, SC34-7029
Java Applications in CICS, SC34-7025

Diagnosis

CICS Problem Determination Guide, GC34-7034
CICS Performance Guide, SC34-7033
CICS Messages and Codes, SC34-7035
CICS Diagnosis Reference, GC34-7038
CICS Recovery and Restart Guide, SC34-7012
CICS Data Areas, GC34-7014
CICS Trace Entries, SC34-7013
CICS Supplementary Data Areas, GC34-7015
CICS Debugging Tools Interfaces Reference, GC34-7039

Communication

CICS Intercommunication Guide, SC34-7018
CICS External Interfaces Guide, SC34-7019

Databases

CICS DB2 Guide, SC34-7011
CICS IMS Database Control Guide, SC34-7016

CICS Shared Data Tables Guide, SC34-7017

CICSPlex SM books for CICS Transaction Server for z/OS

General

CICSPlex SM Concepts and Planning, SC34-7044
CICSPlex SM Web User Interface Guide, SC34-7045

Administration and Management

CICSPlex SM Administration, SC34-7005
CICSPlex SM Operations Views Reference, SC34-7006
CICSPlex SM Monitor Views Reference, SC34-7007
CICSPlex SM Managing Workloads, SC34-7008
CICSPlex SM Managing Resource Usage, SC34-7009
CICSPlex SM Managing Business Applications, SC34-7010

Programming

CICSPlex SM Application Programming Guide, SC34-7030
CICSPlex SM Application Programming Reference, SC34-7031

Diagnosis

CICSPlex SM Resource Tables Reference, SC34-7032
CICSPlex SM Messages and Codes, GC34-7035
CICSPlex SM Problem Determination, GC34-7037

Other CICS publications

The following publications contain further information about CICS, but are not provided as part of CICS Transaction Server for z/OS, Version 4 Release 1.

Designing and Programming CICS Applications, SR23-9692

CICS Application Migration Aid Guide, SC33-0768

CICS Family: API Structure, SC33-1007

CICS Family: Client/Server Programming, SC33-1435

CICS Family: Interproduct Communication, SC34-6853

CICS Family: Communicating from CICS on System/390, SC34-6854

CICS Transaction Gateway for z/OS Administration, SC34-5528

CICS Family: General Information, GC33-0155

CICS 4.1 Sample Applications Guide, SC33-1173

CICS/ESA 3.3 XRF Guide , SC33-0661

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

Readers' Comments — We'd Like to Hear from You

CICS Transaction Server for z/OS
Version 4 Release 1
Debugging Tools Interfaces Reference

Publication No. GC34-7039-01

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: +44 1962 816151
- Send your comments via e-mail to: idrcf@uk.ibm.com

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
GC34-7039-01

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM United Kingdom Limited
User Technologies Department (MP095)
Hursley Park
Winchester
Hampshire
United Kingdom
SO21 2JN

Fold and Tape | **Please do not staple** | Fold and Tape

GC34-7039-01

Cut or Fold
Along Line

GC34-7039-01