CICS Transaction Server for OS/390

IBM

CICSPlex SM Application Progra
Reference

Release 3

SC34-5458-00

CICS Transaction Server for OS/390 IBM

CICSPlex SM Application Progra
Reference

Release 3

SC34-5458-00

Note!

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

First Edition, March 1999

This edition applies to Release 3 of CICS Transaction Server for OS/390, program number 5655-147, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Information in this edition was previously contained in SC33-1430-01, which is now obsolete. Make sure you are using the correct edition for the level of the product.

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

At the back of this publication is a page titled "Sending your comments to IBM". If you want to make comments, but the methods described are not available to you, please address your comments to:

IBM United Kingdom Laboratories, Information Development,
Mail Point 095, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

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

Contents

Notices

Sample programs

Programming interface information

This book is intended to help you write application programs using the CICSPlex SM application programming interface (API). This book documents General-use Programming Interface and Associated Guidance Information provided by CICSPlex SM.

General-use programming interfaces allow the customer to write programs that obtain the services of CICSPlex SM.

Trademarks

Preface

This book provides programming information for the IBM CICSPlex System Manager(CICSPlex SM) element of CICS Transaction Server for OS/390 . It describes how to use the application programming interface (API) to access CICSPlex SM data and services.

Who this book is for

This book is for application programmers who want to access the services of CICSPlex SM.

What you need to know

It is assumed that you have experience writing programs in COBOL, C, PL/I, assembler language, or REXX. You should also have knowledge of the CICSPlex SM concepts and terminology introduced in the *CICSPlex SM Concepts and Planning* book.

For guidance information on how to use the CICSPlex SM API see the *CICSPlex SM Application Programming Guide*.

While you are using this book, you will need to refer to the *CICSPlex SM Resource Tables Reference* for descriptions of the resource tables that you can access. You may also need to refer to the following books:

CICSPlex SM Managing Business Applications
For information about Business Application Services definitions.

CICSPlex SM Managing Resource Usage
For information about real-time analysis and Monitoring definitions.

CICSPlex SM Managing Workloads
For information about Workload Manager definitions.

How to use this book

This book contains reference information about the API commands. Each command description includes:

- A description of what the command does
- The syntax of the command
- A description of the command options in alphabetical order
- A list of the command response values.

Notes on terminology

In the text of this book, the term **CICSPlex SM** (spelled with an uppercase letter 'P') means the IBM CICSPlex System Manager element of CICS Transaction Server for OS/390. The term **CICSplex** (spelled with a lowercase letter 'p') means the largest set of CICS systems to be managed by CICSPlex SM as a single entity.

Other terms used in this book are:

Term	**Meaning**
API	Application programming interface
ASM	Assembler language
CICS TS for OS/390	The CICS element of the CICS TS for OS/390
MVS	MVS/Enterprise Systems Architecture SP (MVS/ESA)

CICS System Connectivity

This release of CICSPlex SM may be used to control CICS systems that are directly connected to it, and indirectly connected through a previous release of CICSPlex SM.

For this release of CICSPlex SM, the directly-connectable CICS systems are:

CICS Transaction Server for OS/390 1.3
CICS Transaction Server for OS/390 1.2
CICS Transaction Server for OS/390 1.1
CICS for MVS/ESA 4.1
CICS Transaction Server for VSE/ESA Release 1
CICS for VSE/ESA 2.3
CICS for OS/2 3.1
CICS for OS/2 3.0

CICS systems that are not directly connectable to this release of CICSPlex SM are:

CICS for MVS/ESA 3.3
CICS for MVS 2.1.2
CICS for VSE/ESA 2.2
CICS/OS2 2.0.1

Note: IBM Service no longer supports these CICS release levels.

You can use this release of CICSPlex SM to control CICS systems that are connected to, and managed by, your previous release of CICSPlex SM. However, if you have any directly-connectable release levels of CICS, as listed above, that are connected to a previous release of CICSPlex SM, you are strongly recommended to migrate them to the current release of CICSPlex SM, to take full advantage of the enhanced management services. See the *CICS Transaction Server for OS/390: Migration Guide* for information on how to do this.

Table 1 shows which CICS systems may be directly connected to which releases of CICSPlex SM.

Table 1. Directly-connectable CICS systems by CICSPlex SM release

CICS system	**CICSPlex SM component of CICS TS 1.3**	**CICSPlex SM 1.3**	**CICSPlex SM 1.2**
CICS TS 1.3	Yes	No	No
CICS TS 1.2	Yes	Yes	No
CICS TS 1.1	Yes	Yes	Yes
CICS for MVS/ESA 4.1	Yes	Yes	Yes
CICS for MVS/ESA 3.3	No	Yes	Yes
CICS for MVS 2.1.2	No	Yes	Yes
CICS TS for VSE/ESA Rel 1	Yes	No	No
CICS for VSE/ESA 2.3	Yes	Yes	Yes
CICS for VSE/ESA 2.2	No	Yes	Yes
CICS for OS/2 3.1	Yes	No	No
CICS for OS/2 3.0	Yes	Yes	Yes
CICS/OS2 2.0.1	No	Yes	Yes

Bibliography

CICS Transaction Server for OS/390

CICS Transaction Server for OS/390: Planning for Installation	GC33-1789
CICS Transaction Server for OS/390: Release Guide	GC34-5352
CICS Transaction Server for OS/390: Migration Guide	GC34-5353
CICS Transaction Server for OS/390: Installation Guide	GC33-1681
CICS Transaction Server for OS/390: Program Directory	GC33-1706
CICS Transaction Server for OS/390: Licensed Program Specification	GC33-1707

CICS books for CICS Transaction Server for OS/390

General	
CICS Master Index	SC33-1704
CICS User's Handbook	SX33-6104
CICS Glossary (softcopy only)	GC33-1705
Administration	
CICS System Definition Guide	SC33-1682
CICS Customization Guide	SC33-1683
CICS Resource Definition Guide	SC33-1684
CICS Operations and Utilities Guide	SC33-1685
CICS Supplied Transactions	SC33-1686
Programming	
CICS Application Programming Guide	SC33-1687
CICS Application Programming Reference	SC33-1688
CICS System Programming Reference	SC33-1689
CICS Front End Programming Interface User's Guide	SC33-1692
CICS C^{++} OO Class Libraries	SC34-5455
CICS Distributed Transaction Programming Guide	SC33-1691
CICS Business Transaction Services	SC34-5268
Diagnosis	
CICS Problem Determination Guide	GC33-1693
CICS Messages and Codes	GC33-1694
CICS Diagnosis Reference	LY33-6088
CICS Data Areas	LY33-6089
CICS Trace Entries	SC34-5446
CICS Supplementary Data Areas	LY33-6090
Communication	
CICS Intercommunication Guide	SC33-1695
CICS Family: Interproduct Communication	SC33-0824
CICS Family: Communicating from CICS on System/390	SC33-1697
CICS External Interfaces Guide	SC33-1944
CICS Internet Guide	SC34-5445
Special topics	
CICS Recovery and Restart Guide	SC33-1698
CICS Performance Guide	SC33-1699
CICS IMS Database Control Guide	SC33-1700
CICS RACF Security Guide	SC33-1701
CICS Shared Data Tables Guide	SC33-1702
CICS Transaction Affinities Utility Guide	SC33-1777
CICS DB2 Guide	SC33-1939

CICSPlex SM books for CICS Transaction Server for OS/390

General	
CICSPlex SM Master Index	SC33-1812
CICSPlex SM Concepts and Planning	GC33-0786
CICSPlex SM User Interface Guide	SC33-0788
CICSPlex SM View Commands Reference Summary	SX33-6099
Administration and Management	
CICSPlex SM Administration	SC34-5401
CICSPlex SM Operations Views Reference	SC33-0789
CICSPlex SM Monitor Views Reference	SC34-5402
CICSPlex SM Managing Workloads	SC33-1807
CICSPlex SM Managing Resource Usage	SC33-1808
CICSPlex SM Managing Business Applications	SC33-1809
Programming	
CICSPlex SM Application Programming Guide	SC34-5457
CICSPlex SM Application Programming Reference	SC34-5458
Diagnosis	
CICSPlex SM Resource Tables Reference	SC33-1220
CICSPlex SM Messages and Codes	GC33-0790
CICSPlex SM Problem Determination	GC33-0791

Other CICS books

CICS Application Programming Primer (VS COBOL II)	SC33-0674
CICS Application Migration Aid Guide	SC33-0768
CICS Family: API Structure	SC33-1007
CICS Family: Client/Server Programming	SC33-1435
CICS Family: General Information	GC33-0155
CICS 4.1 Sample Applications Guide	SC33-1173
CICS/ESA 3.3 XRF Guide	SC33-0661

If you have any questions about the CICS Transaction Server for OS/390 library, see *CICS Transaction Server for OS/390: Planning for Installation* which discusses both hardcopy and softcopy books and the ways that the books can be ordered.

Books from related libraries

Books in related libraries include:

IBM CICS for MVS/ESA Version 4.1

Application Programming Guide, SC33-1169

Application Programming Reference, SC33-1170

System Programming Reference, SC33-1171

IBM CICS TS for OS/390 Release 1

CICS Application Programming Guide, SC33-1687

CICS Application Programming Reference, SC33-1688

CICS System Programming Reference, SC33-1689

CICS Distributed Transaction Programming Guide, SC33-1691

CICS Front End Programming Interface User's Guide, SC33-1692

Please refer to the *CICS Library Guide* for your release of CICS for the titles and form numbers of additional books that support these releases.

TSO/E Version 2

Programming Guide, SC28-1874

Programming Services, SC28-1875

REXX/MVS User's Guide, SC28-1882

REXX/MVS Reference, SC28-1883

NetView Version 2.4

Application Programming Guide, SC31-7081

RODM Programming Guide, SC31-7095

Customization: Using Assembler, SC31-7094

Customization: Using PL/I and C, SC31-7093

Assembler H Version 2

Application Programming Guide, SC26-4036

Application Programming Language Reference, GC26-4037

VS COBOL II Version 1.3.2

Application Programming Guide for MVS and CMS, SC26-4045

Application Programming Guide for VSE, SC26-4697

Application Programming Language Reference, GC26-4047

Application Programming Debugging, SC26-4049

PL/I Version 2.3

Programming Language Reference, SC26-4308

Optimizing Compiler Programmer's Guide, SC33-0006

IBM C/370 Version 2.1

Programming Guide, SC09-1384

Reference Summary, SC09-1211

Summary of Changes

This book is based on information from the CICSPlex SM Release 3 book,*CICSPlex System Manager Application Programming Interface*, SC33-1430-01. The information in this book has been updated to incorporate changes made for CICSPlex SM for CICS Transaction Server for OS/390 Release 3. Changes made since the last edition are indicated by vertical bars to the left of the change.

For further guidance please refer to the *CICSPlex SM Application Programming Guide*.

Chapter 1. Introduction to the commands

This chapter provides standard usage information about the CICSPlex SM application programming interface (API) commands:

| "Using the command-level interface"
| "Using the run-time interface" on page 5

Using the command-level interface

Command format

The format of an API command when issued through the command-level interface is EXECUTE CPSM (or EXEC CPSM) followed by the name of the required command and possibly by one or more options, as follows:

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

where:

`command` Describes the operation required (for example, CONNECT).

`option` Describes any of the required or optional facilities available with each command. Some options are followed by an argument in parentheses. You can write options (including those that require arguments) in any order.

`arg` Which is short for argument, is a value such as *data-value* or *data-ref*. A *data-value* can be a constant. This means that an argument that sends data to CICSPlex SM is generally a *data-value*. An argument that receives data from CICSPlex SM must be a *data-ref*.

Here is an example of an EXEC CPSM command:

```
EXEC CPSM CONNECT
          USER(JONES) VERSION( 14 )
          CONTEXT(EYUPLX 1) SCOPE(EYUCSG 1)
          THREAD(THRDTKN)
          RESPONSE(RESPVAR) REASON(REASVAR)
```

You must add an end-of-command delimiter that is valid for the programming language you are using. In COBOL programs, for example, the end-of-command delimiter is an END-EXEC statement. In PL/I and C programs, the delimiter is a semicolon (;).

Argument values

For the command-level interface, the parenthesized argument values that follow options in an API command are specified as follows:

data-value A sending argument used to pass data from your program to CICSPlex SM.

The data you pass can be fullword binary data, fixed or variable length character data, or unspecified. If the data type is unspecified, CICSPlex SM assumes a composite data structure made up of multiple fields of varying data types.

The argument can be in one of these forms:

Variable name
Self-defining term
Expression.

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

data-ref
: A receiving (or sending and receiving) argument used primarily to pass data from CICSPlex SM to your program.

The data can be any of the same types allowed for *data-value* arguments. However, the argument must be a named variable.

In some cases, you can use a *data-ref* argument to provide input to CICSPlex SM before CICSPlex SM returns its output to you (the COUNT option on the FETCH command is an example of this).

data-area
: A sending or receiving argument used to identify a buffer that contains data. A *data-area* argument can be considered a *data-ref* argument with an unspecified data type. A *data-area* cannot be defined by a self-defining term or expression; it must be a named variable.

ptr-ref
: A receiving argument used to pass pointer values from CICSPlex SM to your program.

A *ptr-ref* argument is a special form of *data-ref* argument. The data being passed is an address pointer, rather than binary or character data.

cpsm-token
: A sending or receiving argument used to pass identifying tokens that are generated by CICSPlex SM. A *cpsm-token* argument can be considered a *data-ref* argument with an unspecified data type.

Tokens are created by CICSPlex SM to identify API processing threads, result sets, filters, and notifications.

Because token values are created by CICSPlex SM, your program must receive a token into a variable before it can specify that token on subsequent commands. A token cannot be defined by a self-defining term or expression; it must be a named variable.

COBOL argument values

The argument values can be replaced as follows:

data-value
: Can be replaced by any COBOL data name of the correct data type for the argument, or by a constant that can be converted to the correct type for the argument. The data type can be specified as one of the following:

Halfword binary — PIC S9(4) USAGE BINARY
Fullword binary — PIC S9(8) USAGE BINARY
Character string — PIC X(n) where “n” is the number of bytes.

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

data-ref
: Can be replaced by any COBOL data name of the correct data type for the argument. The data type can be specified as one of the following:

Halfword binary — PIC S9(4) USAGE BINARY
Fullword binary — PIC S9(8) USAGE BINARY
Character string — PIC X(n) where “n” is the number of bytes.

Where the data type is unspecified, *data-ref* can refer to an elementary or group item.

data-area
: Can be replaced by any COBOL data name with a data type of halfword binary (PIC S9(4) COMP), fullword binary (PIC S9(8) COMP), or character string (PIC X(n)).

ptr-ref
: Can be replaced by a pointer variable or an ADDRESS special register.

cpsm-token
Can be replaced by any COBOL data name with a data type of fullword binary, PIC S9(8) COMP.

C argument values

The argument values can be replaced as follows:

data-value
Can be replaced by any C expression that can be converted to the correct data type for the argument. The data type can be specified as one of the following:

Halfword binary — short int

Fullword binary — long int

Character array — char[n] where “n” is the number of bytes in the field (the field must be padded with blank spaces).

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

data-ref
Can be replaced by any C data reference that has the correct data type for the argument. The data type can be specified as one of the following:

Halfword binary — short int

Fullword binary — long int

Character array — char[n] where “n” is the number of bytes in the field (the field is padded with blank spaces).

If the data type is unspecified, *data-ref* can refer to a scalar data type, array, or structure. The reference must be to contiguous storage.

data-area
Can be replaced by any named variable with a data type of halfword binary (short int), fullword binary (long int), or character array (char[n]).

ptr-ref
Can be replaced by any C pointer type reference.

cpsm-token
Can be replaced by any named variable with a data type of fullword binary, long int.

PL/I argument values

The argument values can be replaced as follows:

data-value
Can be replaced by any PL/I expression that can be converted to the correct data type for the argument. The data type can be specified as one of the following:

Halfword binary — FIXED BIN(15)
Fullword binary — FIXED BIN(31)
Character string — CHAR(n) where “n” is the number of bytes.

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

data-ref
Can be replaced by any PL/I data reference that has the correct data type for the argument. The data type can be specified as one of the following:

Halfword binary — FIXED BIN(15)
Fullword binary — FIXED BIN(31)
Character string — CHAR(n) where “n” is the number of bytes.

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

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

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

data-area
: Can be replaced by any named variable with a data type of halfword binary (FIXED BIN(15)), fullword binary (FIXED BIN(31)), or character string (CHAR(n)).

ptr-ref
: Can be replaced by any PL/I reference of type POINTER ALIGNED.

cpsm-token
: Can be replaced by any named variable with a data type of fullword binary, FIXED BIN(31).

Assembler language argument values

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

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

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

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

The argument values can be replaced as follows:

data-value
: Can be replaced by a relocatable expression that is an assembler-language reference to data of the correct type for the argument, or by a constant of the correct type for the argument.

data-ref
: Can be replaced by a relocatable expression that is an assembler-language reference to data of the correct type for the argument.

data-area
: Can be replaced by a relocatable expression that is an assembler-language reference to data with a type of halfword (DS H), fullword (DS F), or character string (CLn).

ptr-ref
: Can be replaced by any absolute expression that is an assembler-language reference to a register.

cpsm-token
: Can be replaced by a relocatable expression that is an assembler-language reference to data with a type of fullword, DS F.

Using the run-time interface

Command format

An API command can be passed from REXX to CICSPlex SM in one of two ways. The first method is to use the REXX ADDRESS command, like this:

```
ADDRESS CPSM 'command option(arg)...'
```

This method of calling the API invokes a CICSPlex SM host subcommand environment.

Alternatively, you can use the EYUAPI() function supplied by CICSPlex SM:

```
var = EYUAPI('command option(arg)...')
```

This method invokes the CICSPlex SM REXX function package.

Note that with both methods you can enter text in either upper or lower case.

Here is an example of an API command as it would be issued from a REXX program:

```
var = EYUAPI('CONNECT'             ,
            'CONTEXT('WCONTEXT')' ,
            'SCOPE('WSCOPE')'     ,
            'VERSION( 14 )'      ,
            'THREAD(THRDTKN)'     ,
            'RESPONSE(RESPVAR)' ,
            'REASON(REASVAR)')
   .
   .
```

Argument values

The CICSPlex SM run-time interface makes full use of the standard REXX variable interface. REXX processes variables differently depending on the parameter's data type and whether it is used for input, output, or both. In addition, REXX provides substitution of variables into a command stream that may in some cases make them transparent to the run-time interface.

For the REXX run-time interface, the parenthesized argument values that follow options in an API command are specified as follows:

data-value A sending argument used to pass character or binary data from your program to CICSPlex SM.

A *data-value* argument is considered to be character input. Binary data (including EYUDA and CVDA values) is translated into the appropriate internal format. User tokens are not translated.

data-ref A receiving (or sending and receiving) argument used primarily to pass data from CICSPlex SM to your program.

A *data-ref* argument must be a named variable that can be used to receive the resulting output. The output data is translated as appropriate:

Character data is not translated; the data is placed into the variable as is.

Binary data is translated to display format (decimal) and placed into the variable.

User tokens are not translated; the token value is placed into the variable as is.

Address values are not translated; the specified storage buffer is placed directly into one or more variables.

In some cases, you can use a *data-ref* argument to provide input to CICSPlex SM before CICSPlex SM returns its output to you (the COUNT option on the FETCH command is an example of this). If a *data-ref* argument can be supplied as input, you must specify a variable for that argument. If you do not want to specify an input value, you should initialize the variable.

data-area
: A sending or receiving argument used to identify a buffer that contains data. A *data-area* argument must be a named variable.

For output buffers that could receive multiple resource table records, CICSPlex SM creates (or fills) stem variables to hold the data. The zero entry of the stem array indicates the number of entries in the array.

For example, in the stem variable called W_INTO_EVALDEF, the W_INTO_EVALDEF.0 entry contains the number of EVALDEF resource table records returned. The entries W_INTO_EVALDEF.1 through W_INTO_EVALDEF.n contain the actual resource table records.

A stem variable is created regardless of whether the actual output is a single record or multiple records.

ptr-ref
: A receiving argument used to pass pointer values from CICSPlex SM to your program.

A *ptr-ref* argument must be a named variable that can be used to receive the resulting output. The data being passed is a character representation of a hexadecimal address.

cpsm-token
: A sending or receiving argument used to pass identifying tokens that are generated by CICSPlex SM.

A *cpsm-token* argument must be a named variable. Tokens are not translated; the token value is placed into the variable as is.

Note: Each variable (or stem variable) returned by CICSPlex SM contains an entire resource table record. You can use the TPARSE command to break a record into individual fields. For a description of this command, see Chapter 3, “REXX functions and commands” on page 87.

Syntax notation used in this book

In this book, the CICSPlex SM API commands are presented in a standard way.

The EXEC CPSM that precedes the command name in the command-level interface is not shown, nor is the end-of-command delimiter. Likewise, the ADDRESS CPSM or var=EYUAPI() that is required for the REXX run-time interface is not shown.

You interpret the syntax diagrams shown in this book by following the arrows from left to right. The conventions are:

Symbol	Meaning
A B C	A set of mutually exclusive alternatives, one of which you *must* code.
A B C	A set of mutually exclusive alternatives, one of which you *may* code.
A B C	A set of alternatives, any number of which you may code.
A B	Alternatives where A is the default.
Name **Name:** A B	See the separate syntax fragment whose name is shown.
Punctuation and uppercase characters	Code exactly as shown.
Lowercase italics	Code your own text, as appropriate (for example, *name*).

For example, with `CONNECT VERSION(data-value)` you must code `CONNECT VERSION` and `()` as they appear, but are free to code any four-character number that represents a valid release of CICSPlex SM.

MVS/ESA restrictions

The following general restrictions apply to all CICSPlex SM API commands:

The program must be in primary addressing mode when invoking any CICSPlex SM service. The primary address space must be the home address space. All parameters passed to CICSPlex SM must reside in the primary address space.

CICSPlex SM does not always preserve access registers across commands. If your program uses access registers, it should save them before invoking a CICSPlex SM service, and restore them before reusing them.

Language considerations

All of the language considerations that apply to the various environments (CICS, MVS/ESA batch, TSO, and NetView) also apply to CICSPlex SM programs written to run in those environments.

CICS and CICSPlex SM value data areas

The values for some CICSPlex SM resource table attributes are maintained in an encoded form. These values can be:

CICSPlex SM value data areas (EYUDAs)
CICS value data areas (CVDAs).

You can use one of two built-in translator functions to translate these values:

EYUDAs Use the CICSPlex SM translator function called EYUVALUE. You must also specify the CPSM translator option when you run the CICS/ESA translator.

CVDAs Use the CICS translator function called DFHVALUE. You must also specify the CICS translator option when you run the CICS/ESA translator.

For example, consider the following COBOL statement:

```
MOVE EYUVALUE(QUIESCING) TO EYUDATA
```

This statement translates the EYUDA character value of QUIESCING into its numeric equivalent of 48 when the program is translated.

Notes:

1. The EYUVALUE function is not available to programs written in REXX. You can use the TPARSE command, which is supplied specifically for REXX programs, to access and translate the attribute values in a resource table. For a description of this command, see Chapter 3, “REXX functions and commands” on page 87.
2. In some CICS environments, the DFHVALUE function returns incompatible CVDA values for the following resource table attributes:

Resource table	Attribute value	CICS Environment
CONNECT	RECOVSTATUS(NRS)	CICS/VSE
LOCTRAN	RESSEC(RESSECEXT)	CICS/VSE CICS/MVS
PROGRAM	LPASTAT(SVA)	CICS/VSE
PROGRAM	LPASTAT(NOTSVA)	CICS/VSE

   Because these CVDA values conflict with values used in other CICS environments, CICSPlex SM must modify them to retain their uniqueness. CICSPlex SM adds 9000 to the value returned by DFHVALUE for each of these CICS CVDA attributes.

CICSPlex SM also provides a TRANSLATE command to translate EYUDA and CVDA values at run time. You can use TRANSLATE to convert an EYUDA or CVDA value that is associated with a specific resource table and attribute. For example:

```
EXEC CPSM TRANSLATE OBJECT(WLMAWAOR)
                    ATTRIBUTE(STATUS)
                    FROMCV(48)
                    TOCHAR(EYUCHAR)
                    RESPONSE(RESPDATA)
                    REASON(REASDATA)
```

This command translates the EYUDA value for the STATUS attribute of the WLMAWAOR resource table into its character value when the program is run.

For a description of the TRANSLATE command, see “TRANSLATE” on page 79.

Note: For a list of the EYUDA values used by CICSPlex SM, see Appendix B, “EYUDA values” on page 99.

Length options

Many API commands involve the transfer of data between the application program and CICSPlex SM.

In VS COBOL II, PL/I, and Assembler language, the translator can default certain length options; this means they may be optional in programs that specify data areas. In C and REXX, all length options must be specified.

The CICSPlex SM API allows most `data-value` arguments, which are only passed from your program to CICSPlex SM, to default. The exception is the LENGTH option on the following commands:

CREATE
REMOVE

UPDATE

On the other hand, `data-ref` arguments, which can be passed from your program to CICSPlex SM and back again, must always be specified.

When an API command offers a length option, it is always expressed as a signed fullword binary value. This puts a theoretical upper limit of 2 147 483 647 bytes on the length. The achievable upper limit varies from command to command and with various language compilers, but it is somewhat less than the theoretical maximum.

RESPONSE and REASON options

Once an API command completes processing, it returns a response and, if appropriate, a reason. You must specify the RESPONSE and REASON options on each command to receive the response and reason values returned by that command.

Note: The TBUILD and TPARSE commands, which can be used only with the REXX run-time interface, do not use the RESPONSE and REASON options. The result of these REXX-specific processes is returned by their STATUS option. For more information, see the descriptions of the TBUILD and TPARSE commands in Chapter 3, “REXX functions and commands” on page 87.

RESPONSE*(data-ref)*
data-ref is a user-defined variable. On return from the command, it contains a character value that describes the result of command processing. RESPONSE values are given in the description of each command.

REASON*(data-ref)*
data-ref is a user-defined variable. On return from the command, it contains a value that further qualifies the response to certain commands. REASON values are given with the RESPONSE values, for those responses that use them.

For more information about the RESPONSE and REASON options, see *CICSPlex SM Application Programming Guide*. For a summary of RESPONSE and REASON values by command, see Appendix A, “RESPONSE and REASON values” on page 93.

Chapter 2. The API commands

This chapter contains detailed descriptions of the CICSPlex SM API commands. All of these commands can be used with either the command-level interface or the REXX run-time interface.

Each description includes the following, as appropriate:

- A description of the command
- Usage notes
- Related commands
- Syntax of the command
- Available options for the command
- Responses returned by the command

The commands are presented in alphabetical order:

ADDRESS

Provide access to CICSPlex SM storage areas.

```
      ADDRESS                                          THREAD(cpsm-token)  RESPONSE(data-ref)
               ECB(ptr-ref)      SENTINEL(ptr-ref)
     REASON(data-ref)
```

Description

The ADDRESS command provides access to CICSPlex SM storage areas.

ADDRESS returns the addresses of two control fields that are associated with each API thread:

- the event control block (ECB)
- the sentinel.

If your program is written in REXX, the ECB and sentinel values are returned as character representations of the hexadecimal addresses. You have to use the REXX STORAGE function to access the storage at those addresses.

Related commands

LISTEN, RECEIVE

Options

ECB*(ptr-ref)*
Names a variable to receive the address of the ECB that will be posted when asynchronous requests associated with this thread are awaiting processing. The ECB field is cleared whenever the counter value in the SENTINEL field reaches 0.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

SENTINEL*(ptr-ref)*
Names a variable to receive the address of a 4-byte counter of completed asynchronous requests associated with this thread.

The sentinel value increases each time an asynchronous request completes. Examples of asynchronous requests include:

A command is issued with the NOWAIT option
An event occurs that is named in a LISTEN command.

The sentinel value decreases when a RECEIVE command is issued. If the counter value is 0, it means there are no outstanding asynchronous requests to be received.

Note: Each API processing thread can handle a maximum of 256 asynchronous requests (as indicated by the SENTINEL counter) at one time.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the ADDRESS command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

ECB
SENTINEL
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

CANCEL

Cancel the notification request produced by a previous LISTEN command.

```
     CANCEL  NOTIFICATION(cpsm-token)  THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command cancels the notification request produced by a previous LISTEN command.

Related commands

LISTEN

Options

NOTIFICATION*(cpsm-token)*
Identifies the notification request to be cancelled. The *cpsm-token* value that identifies a notification request is returned by the LISTEN command.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the CANCEL command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

NOTIFICATION
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

CONNECT

Establishe a connection with CICSPlex SM, defines an API processing thread, and provides default settings to be used by the thread.

```
      CONNECT  VERSION(data-value)
                             USER(data-value)
                                          SIGNONPARM(data-value)
                                             THREAD(data-ref)  RESPONSE(data-ref)
      CONTEXT(data-value)
                        SCOPE(data-value)
     REASON(data-ref)
```

Description

The specifics of the connection process depend upon the environment in which your program is running. For a complete description of the connection process, see *CICSPlex SM Application Programming Guide*.

Related commands

DISCONNECT, QUALIFY, TERMINATE

Options

CONTEXT(*data-value*)
Identifies the default context for commands issued against this thread. The context must be the 1- to 8-character name of a CMAS or CICSplex.

The default context is in effect for all commands issued against the thread unless you override it for a specific command or change it by issuing the QUALIFY command. As an alternative to specifying a default context for the thread, you can specify the context for individual commands as they are processed.

If you do not specify the CONTEXT option, the default context for the thread is the CMAS to which the thread is connected.

REASON(*data-ref*)
Names a variable to receive the fullword reason value returned by this command.

RESPONSE(*data-ref*)
Names a variable to receive the fullword response value returned by this command.

SCOPE(*data-value*)
Identifies the default scope for commands issued against this thread.

The SCOPE option qualifies the CONTEXT option. When the context is a CICSplex, the scope can be:

The 1- to 8-character name of the CICSplex itself
A CICS system or CICS system group within the CICSplex
A logical scope, as defined in a CICSPlex SM resource description (RESDESC).

When the context is a CMAS, this option has no meaning and is ignored.

The default scope is in effect for all commands issued against the thread unless you override it for a specific command or change it by issuing the QUALIFY command. If you do not specify the SCOPE option, no default scope is assumed.

Note: Certain API commands require a valid scope when the context is a CICSplex. If you do not specify a scope on a CONNECT or QUALIFY command, then you must specify the SCOPE option when you issue any of these commands for a resource table that represents a CICS resource:

GET
PERFORM OBJECT
PERFORM SET
REFRESH
SET.

SIGNONPARM(*data-value*)
Identifies a 1- to 8-character signon parameter to be passed to the API security exit routine (EYU9XESV) at your enterprise.

If CMAS security is active and CICSPlex SM finds no security defined in the environment where the API program is running, it passes the USER and SIGNONPARM values from the CONNECT command to EYU9XESV. For more information about API security, see *CICSPlex SM Application Programming Guide*.

THREAD(*data-ref*)
Names a variable to receive the fullword token that CICSPlex SM assigns to this processing thread.

This identifying token must be specified on all subsequent commands issued against this thread.

USER(*data-value*)
Identifies a 1- to 8-character user ID to be passed to the API security exit routine (EYU9XESV) at your enterprise.

If CMAS security is active and CICSPlex SM finds no security defined in the environment where the API program is running, it passes the USER and SIGNONPARM values from the CONNECT command to EYU9XESV. For more information about API security, see *CICSPlex SM Application Programming Guide*.

VERSION*(data-value)*
Identifies the release of CICSPlex SM resource table data that you want to be available to your program. The VERSION value must be the 4-character number of a valid CICSPlex SM release, such as 0140 for CICS Transaction Server for OS/390 Release 3.

Notes:

1. The VERSION value must be 0120 or greater. The API cannot access data from a release of CICSPlex SM earlier than Release 2.
2. The VERSION value must be less than or equal to the version of the CICSPlex SM run-time environment.
3. You can specify a VERSION value that is greater than the release under which your API program was originally written, provided:

   You compile your program using the appropriate copy books for the version specified.
   Your program is compatible with the copy books for the version specified.

   For complete details on things to consider when running under a different release, see *CICSPlex SM Application Programming Guide*.

Conditions

The following is a list of the RESPONSE values that can be returned by the CONNECT command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

ENVIRONERROR
An environment error occurred for one of the following reasons:

APITASKERR	The API control subtask encountered an error during startup.
NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.
SOERESOURCE	A required resource that is owned by the Environment Services System Services (ESSS) address space is not available.
SOLRESOURCE	A required resource that is locally owned (that is, owned by the address space where the processing thread is running) is not available. |

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

CONTEXT
SCOPE
SIGNONPARM
USRID
VERSION.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.
CPSMSERVER	The CMAS to which the processing thread was trying to connect is not available.
CPSMSYSTEM	No CICSPlex SM systems are available.
CPSMVERSION	No CICSPlex SM system at the specified version is available.

NOTPERMIT
A not permitted condition occurred for one of the following reasons:

EXPIRED	The security authorization of the specified user ID has expired.
SIGNONPARM	The specified signon parameter is not authorized for the user ID.
USRID	The specified user ID does not have the required security authorization.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.

COPY

Copy resource table records.

```
                                    ALL                                        REPLACE
      COPY  FROM(cpsm-token)  TO(cpsm-token)
                                    CURRENT                COUNT(data-ref)       ADD
                                    FILTER(cpsm-token)                         UPDATE
                                    MARKED
                                    NOTFILTER(cpsm-token)
                                    NOTMARKED

     THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command copies some or all of the resource table records in one result set to another result set on the same processin thread.

The COPY command always begins processing with the last record that was fetched, rather than the next one in the result set.

The target result set can be an existing result set or a new one that is created by this process. If you specify an existing result set as the target, you can either overwrite the existing records or add to them.

A result set can contain only one record for a given resource. If duplicate records are found during the copy process, the ADD, REPLACE or UPDATE option you specified determines which record is retained.

To copy selected records from a source result set, you can use:

– The SPECIFY FILTER command to define a filter for the source result set.

– The MARK and UNMARK commands to mark records in the source result set. Any marks you place on records in the source result set are not retained when those records are copied to the target result set.

The relative position of records in the target result set may not be the same as it was in the source result set. The position can be affected by:

– Deleted records being left in the source result set (when COPY ALL is specified) and other records assuming their position in the target result set.

– The sort order associated with the target result set, if any. If the target result set does not exist, records are copied in the same order as they appeared in the source result set. If an existing result set is named as the target, records are copied and then sorted according to the sort order that was in effect for that result set.

Related commands

DELETE, DISCARD, GET, GETDEF, LOCATE, MARK, ORDER, PERFORM OBJECT, QUERY, SPECIFY FILTER

Options

ADD
Adds the resource table records from the source result set to an existing target result set. If duplicate records are found, the record in the target result set is retained.

If no existing result set is specified as the target, the ADD option is ignored.

ALL
Copies all the resource table records in the source result set to the target result set.

Any records that have been deleted from the source result set are not copied. In effect, the ALL option compresses a result set by leaving deleted records in the source result set and copying the remaining records to a new result set.

COUNT*(data-ref)*
Names a variable to receive the number of resource table records in the target result set after the copy process is complete.

CURRENT
Copies only the current resource table record in the source result set to the target result set.

FILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The FILTER option copies only those resource table records that meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

FROM*(cpsm-token)*
Identifies the source result set for this operation. The result set can be one produced by any of these commands:

COPY
GET

GETDEF
PERFORM OBJECT.

MARKED
Copies only those resource table records that are marked in the source result set. You can mark resource table records by using the MARK and UNMARK commands.

NOTFILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The NOTFILTER option copies only those resource table records that do not meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

NOTMARKED
Copies only those resource table records that are not marked in the source result set. You can mark resource table records by using the MARK and UNMARK commands.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

REPLACE
Deletes the resource table records in an existing target result set and replaces them with the results of this copy operation. If the copy operation does not result in any resource table records being copied, the target result set is discarded.

If no existing result set is specified as the target, the REPLACE option is ignored.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

TO*(cpsm-token)*
Identifies the target result set for this operation. The result set can be one produced by any of these commands:

COPY
GET
GETDEF
PERFORM OBJECT.

Note: The target result set cannot be the same as the source result set that you specified on the FROM option.

If this field is:

Set to binary zero (in COBOL, C, PL/I or Assembler)
An uninitialized variable (in REXX).

CICSPlex SM creates a new result set and returns its identifying token in the same field.

UPDATE
Updates an existing target result set with resource table records from the source result set. If duplicate records are found, the record in the source result set replaces the record in the target result set.

If no existing result set is specified as the target, the UPDATE option is ignored.

Conditions

The following is a list of the RESPONSE values that can be returned by the COPY command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

NODATA
No records were found that matched the specified search criteria.

BUSY
A busy condition occurred for one of the following reasons:

FROM	The source result set specified on the FROM option is being processed by another command.
TO	The target result set specified on the TO option is being processed by another command. This condition can occur if you specified the same result set on the FROM and TO options.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
| **SOCRESOURCE**	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INCOMPATIBLE
An incompatible condition occurred for one of the following reasons:

INVALIDOBJ	The target result set specified on the TO option is not compatible with the source result set specified on the FROM option. The result sets must contain the same type of resource table records.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

FILTER
FROM
NOTFILTER
THREAD
TO.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

CREATE

Create a new CICSPlex SM or CICS definition.

```
      CREATE  OBJECT(data-value)  FROM(data-area)  LENGTH(data-value)
                                                                         THREAD(cpsm-token)
       PARM(data-area)  PARMLEN(data-value)      CONTEXT(data-value)
     RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command creates a new CICSPlex SM or CICS definition using the attribute values you specify. The new definition is stored in the CICSPlex SM data repository. For definitions that have a CICSplex as their context (such as workload management or real-time analysis definitions), the new definition is automatically distributed to all the CMASs involved in managing the CICSplex.

Related commands

REMOVE, UPDATE

Options

CONTEXT*(data-value)*
Identifies the context for this command. The context must be the 1- to 8-character name of a CMAS or CICSplex.

If you do not specify the CONTEXT option, the default context for the thread is assumed.

FROM*(data-area)*
Identifies a buffer containing a resource table record that represents the definition to be created.

The record must include all of the attributes for the resource table specified on the OBJECT option. For optional attributes that you do not want to specify, set the field to the appropriate null value for the attribute's data type.

LENGTH*(data-value)*
A fullword value that specifies the length of the FROM buffer.

OBJECT*(data-value)*
Identifies the resource table that represents the definition being created. This value must be the 1- to 8-character name of a valid CPSM Definition or CICS Definition resource table. For a list of the CICSPlex SM resource tables by type, see *CICSPlex SM Application Programming Guide*.

PARM*(data-area)*
Identifies a buffer containing the parameter expression to be used in creating the definition.

For details on how to use a parameter expression with the CREATE command, see *CICSPlex SM Application Programming Guide*. For a description of the parameters that are valid for a given resource table, see *CICSPlex SM Resource Tables Reference*.

PARMLEN*(data-value)*
A fullword value that specifies the length of the PARM buffer.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the CREATE command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
REQTIMEOUT	One of the CMASs to which the request was directed did not respond.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

CONTEXT
FROM
LENGTH
OBJECT
PARM
PARMLEN
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CMAS	A CMAS to which the request was directed is not available.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.
MAINTPOINT	The maintenance point for the current context is not available.

NOTPERMIT
A not permitted condition occurred for one of the following reasons:

USRID	The user ID associated with the processing thread does not have the required security authorization.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

TABLEERROR
A resource table record is invalid for one of the following reasons:

DATAERROR	The value associated with one or more resource table attributes is invalid. This error can occur if: The resource table is missing required attributes, contains one or more conflicting attributes, or is a duplicate. A CICS resource definition contains attributes that would cause the EXEC CICS CREATE command to issue warnings. Use the FEEDBACK command to retrieve additional data about this error.
INVALIDATTR	One of the resource table attributes is invalid.
INVALIDVER	The specified version of the resource table is not supported by CICSPlex SM.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

DELETE

Delete resource table records.

```
          CURRENT
   DELETE                                                   RESULT(cpsm-token)  THREAD(cpsm-token)
          ALL                       COUNT(data-ref)
          FILTER(cpsm-token)
          MARKED
          NOTFILTER(cpsm-token)
          NOTMARKED
  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command deletes one or more resource table records from a result set.

The DELETE command always begins processing with the last record that was fetched, rather than the next one in the result set.

The records you delete are marked as deleted, but they retain their positions in the result set. The remaining records also retain their positions; they are not renumbered. Any API commands that you issue after a DELETE command skip over the deleted records in a result set. One exception is the ORDER command, which sorts all the records in a result set, including deleted records. If you try to issue a command against a deleted record, you receive a RESPONSE value of NODATA.

To remove deleted records and compress a result set, you can copy the remaining records to a new result set. Use the COPY command with the ALL option to copy all the records in a result set except those that have been deleted.

Note: Deleted records are also removed and the remaining records renumbered when you issue a REFRESH command.

Related commands

COPY, DISCARD, GET, GROUP, LOCATE, MARK, ORDER, PERFORM OBJECT, REFRESH, SPECIFY FILTER

Options

ALL
Deletes all the resource table records in the result set.

COUNT*(data-ref)*
Names a variable to receive the number of resource table records in the result set after the delete process is complete.

CURRENT
Deletes only the current resource table record in the result set.

Note: The record pointer remains positioned on the deleted record. If you issue another API command with the CURRENT option before repositioning the pointer, you receive a RESPONSE value of NODATA.

FILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The FILTER option deletes only those resource table records that meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

MARKED
Deletes only those resource table records that are marked in the result set. You can mark resource table records by using the MARK and UNMARK commands.

NOTFILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The NOTFILTER option deletes only those resource table records that do not meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

NOTMARKED
Deletes only those resource table records that are not marked in the result set. You can mark resource table records by using the MARK and UNMARK commands.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY

GET
GROUP
PERFORM OBJECT.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the DELETE command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

NODATA
No records were found that matched the specified search criteria.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.
SOLRESOURCE	A required resource that is locally owned (that is, owned by the address space where the processing thread is running) is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

FILTER
NOTFILTER
RESULT
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

DISCARD

Discard a result set, filter, or view.

```
      DISCARD    FILTER(cpsm-token)    THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
                RESULT(cpsm-token)
                VIEW(data-value)
```

Description

This command discards a result set, filter, or view.

Related commands

COPY, GET, GETDEF, GROUP, PERFORM OBJECT, SPECIFY FILTER, SPECIFY VIEW

Options

FILTER*(cpsm-token)*
Identifies the filter to be discarded. The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
Identifies the API result set to be discarded. The result set can be one produced by any of these commands:

COPY
GET
GETDEF
GROUP
PERFORM OBJECT.

Note: If you discard a result set that was summarized by the GROUP command, all of the summarized result sets are also discarded.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

VIEW*(data-value)*
Identifies the view to be discarded. This value must be the 1- to 8-character name of a view as defined on a SPECIFY VIEW command.

Conditions

The following is a list of the RESPONSE values that can be returned by the DISCARD command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INUSE
An in use condition occurred for one of the following reasons:

FILTER	The specified filter is currently in use and cannot be discarded.
VIEW	The specified view is currently in use and cannot be discarded.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

FILTER
RESULT
THREAD
VIEW.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

DISCONNECT

Disconnects an API processing thread from CICSPlex SM.

```
      DISCONNECT  THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
```

Description

Any resources that are associated with the thread are released, including result sets, filters, views, diagnostic data, and outstanding asynchronous requests.

Related commands

CONNECT, TERMINATE

Options

REASON(*data-ref*)
Names a variable to receive the fullword reason value returned by this command.

RESPONSE(*data-ref*)
Names a variable to receive the fullword response value returned by this command.

THREAD(*cpsm-token*)
Identifies the API thread to be disconnected. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the DISCONNECT command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

FEEDBACK

Retrieve diagnostic data.

```
      FEEDBACK  INTO(data-area)  LENGTH(data-ref)
                                                          NEXT
                                              RESULT(cpsm-token)
                                                          FIRST     COUNT(data-ref)
     THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command retrieves diagnostic data about a previously issued API command.

The diagnostic data is returned as FEEDBACK resource table records.

If the previous command involved processing a result set and it returned a RESPONSE value other than OK, a FEEDBACK resource table record is appended to the end of each resource table record in the result set that had an error associated with it causing the non-OK RESPONSE to be sent. The diagnostic data is available to the FEEDBACK command until another command processes the same result set. At that point, the data is replaced with FEEDBACK records for the subsequent command.

Note: No FEEDBACK records are produced if a command that processed a result set returned a RESPONSE value of OK.

If the previous command did not process a result set, the FEEDBACK resource table records are returned in a separate feedback area. The records in that feedback area are cleared and refreshed for each command that is not result set-oriented. So for commands that place their diagnostic data in the feedback area rather than in a result set, FEEDBACK can retrieve data only for the most recently issued command.

Once you have issued the FEEDBACK command to retrieve diagnostic data for a command, the feedback record or area is cleared. You cannot request the same FEEDBACK resource table records more than once.

If a command is processed asynchronously (that is, you specify the NOWAIT option) the diagnostic data for that command is returned in the ASYNCREQ notification resource table. No FEEDBACK resource table records are produced for an asynchronous request.

Diagnostic data is not available for these commands:

- DISCONNECT
- FEEDBACK
- TERMINATE

The TBUILD and TPARSE commands supplied for use in REXX programs do not provide any useful FEEDBACK information.

For a complete description of the FEEDBACK resource table, see *CICSPlex SM Resource Tables Reference*.

Options

COUNT*(data-ref)*
Specifies the number of feedback records to be retrieved from the result set named in the RESULT option. If you do not specify the COUNT option, only one feedback record is retrieved.

If you are retrieving multiple feedback records, they are placed one after another in the INTO buffer. The INTO buffer must be long enough to hold all the feedback records being retrieved.

The value that CICSPlex SM returns in this field depends on the RESPONSE value for the FEEDBACK command:

OK The actual number of records returned in the INTO buffer.

WARNING AREATOOSMALL
The number of records returned in the INTO buffer, which is not the total number of records requested.

INVALIDPARM LENGTH
The field is not set because the INTO buffer was not long enough to hold even one resource table record.

FIRST
Retrieves the first feedback record from the result set named in the RESULT option.

If you specify the COUNT option, FIRST retrieves the specified number of records, beginning with the first record in the result set.

INTO*(data-area)*
Identifies a buffer (or stem variable, in REXX) to receive the feedback data. This buffer must be long enough to hold all the feedback data being retrieved.

LENGTH*(data-ref)*
A fullword value that specifies the length of the INTO buffer.

The value that CICSPlex SM returns in this field depends on the RESPONSE value for the FEEDBACK command:

OK The actual length of the data returned in the INTO buffer.

WARNING AREATOOSMALL
The buffer length that would be required to hold all the requested records.

INVALIDPARM LENGTH
The field is not set because the INTO buffer was not long enough to hold even one resource table record.

NEXT
Retrieves the next available feedback record from the result set named in the RESULT option.

If you specify the COUNT option, NEXT retrieves the specified number of records, beginning with the next record in the result set.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
Identifies an API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
GROUP
PERFORM OBJECT.

Use the RESULT option to retrieve feedback data about a previously issued command that processed a result set. Use FEEDBACK without the RESULT option to retrieve data about the most recently issued command that did not process a result set.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the FEEDBACK command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

NODATA
No records were found that matched the specified search criteria, or a command that processed a result set returned a RESPONSE of OK.

WARNING
The command completed processing with a warning, for the following reason:

AREATOOSMALL	The INTO buffer is not long enough to hold the number of records requested and available.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

COUNT
INTO
LENGTH
RESULT
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

FETCH

Retrieve data and status information for resource table records.

```
      FETCH  INTO(data-area)  LENGTH(data-ref)
                                   ALL                     POSITION(data-value)
                                   FILTER(cpsm-token)
                                   MARKED
                                   NOTFILTER(cpsm-token)
                                   NOTMARKED
                     DATA                FORWARD
                                                        RESULT(cpsm-token)  THREAD(cpsm-tok
       COUNT(data-ref)      BOTH        DETAIL       BACKWARD
                     STATUS
      RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command retrieves data and status information for one or more resource table records in a result set.

After a FETCH command, the record pointer is usually positioned to the next record in the result set (that is, the record following the last record fetched in whichever direction the pointer was moving, forward or backward). However, the following API commands always act upon the last record that was fetched (that is, the record pointer is not advanced):

– COPY
– DELETE
– MARK.
– UNMARK

If no records were fetched (because no records matched the specified criteria), the pointer is positioned to the top or bottom of the result set, depending on which direction it was moving.

Related commands

COPY, GET, GETDEF, GROUP, LOCATE, MARK, ORDER, PERFORM OBJECT, QUERY, SPECIFY FILTER

Options

ALL
Retrieves all the resource table records in the result set. When you specify ALL, the POSITION and COUNT options are ignored.

BACKWARD
Begins the retrieval process with the last record fetched and continues in a backward direction through the specified result set.

BOTH
Retrieves both the resource table data and the OBJSTAT status information about the last action performed against the resource table. Each record contains OBJSTAT information followed by resource table data.

COUNT_(data-ref)_
Specifies the number of resource table records to be retrieved.

The COUNT option applies to the result set named in the RESULT option, unless you also specify the DETAIL option. When you specify DETAIL, COUNT applies to the source records associated with the summarized result set named in RESULT.

If you do not specify the COUNT option, only one record is retrieved in most cases. However, if you do not specify COUNT when you specify DETAIL, all the source records associated with the current summary resource table record are retrieved.

If you are retrieving multiple records, they are placed one after another in the INTO buffer. The INTO buffer must be long enough to hold all the records being retrieved.

The value that CICSPlex SM returns in this field depends on the RESPONSE value for the FETCH command:

OK The actual number of records returned in the INTO buffer.

WARNING AREATOOSMALL
The number of records returned in the INTO buffer, which is not the total number of records requested.

INVALIDPARM LENGTH
The field is not set because the INTO buffer was not long enough to hold even one resource table record.

DATA
Retrieves only the specified resource table data. The records do not contain any OBJSTAT status information about the last action performed against the resource table.

Note: The OBJSTAT information includes a summary count field that is set when resource table records are summarized using the GROUP command. If you plan to GROUP the resource table records and you want to know how many records are combined to form a summary record, you should specify BOTH to obtain both data and OBJSTAT information when the records are fetched.

DETAIL
Retrieves the source records associated with a specific summary resource table record.

When you specify DETAIL, the result set named on the RESULT option must be a summarized result set. DETAIL expands the summary record by retrieving the resource table records associated with it from the source result set. If you do not specify DETAIL when a summarized result set is being processed, the summary records themselves are retrieved. If the result set is not a summarized result set, this option has no meaning and is ignored.

You can use the FORWARD or BACKWARD option with DETAIL to select which summary record you want to expand. The FORWARD and BACKWARD options also control the direction in which records are retrieved from the source result set.

By default, all the source records associated with the summary record are retrieved. However, you can use the COUNT, FILTER, or NOTFILTER, option to limit the records retrieved from the source result set. You can also use the MARKED or NOTMARKED option to retrieve only those records associated with the summary record that are marked (or not marked) in the source result set.

Note: You cannot explicitly position the record pointer in the source result set. When you specify DETAIL, the POSITION option is ignored.

For more information on processing summarized result sets, see *CICSPlex SM Application Programming Guide*. For a description of the GROUP command, which creates summarized result sets, see "GROUP" on page 41.

FILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The FILTER option indicates that only those resource table records that meet the specified filter criteria should be considered for retrieval. The number of records that are actually retrieved is determined by the COUNT option.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

FORWARD
Begins the retrieval process with the next record (that is, the record that follows the last record fetched) and continues in a forward direction through the specified result set.

INTO*(data-area)*
Identifies a buffer (or stem variable, in REXX) to receive the resource table records. This buffer must be long enough to hold all the records being retrieved.

LENGTH*(data-ref)*
A fullword value that specifies the length of the INTO buffer.

The value that CICSPlex SM returns in this field depends on the RESPONSE value for the FETCH command:

OK The actual length of the data returned in the INTO buffer.

WARNING AREATOOSMALL
The buffer length that would be required to hold all the requested records.

INVALIDPARM LENGTH
The field is not set because the INTO buffer was not long enough to hold even one resource table record.

MARKED
Indicates that only those resource table records that are marked in the result set should be considered for retrieval. The number of records that are actually retrieved is determined by the COUNT option.

You can mark resource table records by using the MARK and UNMARK commands.

NOTFILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The NOTFILTER option indicates that only those resource table records that do not meet the specified filter criteria should be considered for retrieval. The number of records that are actually retrieved is determined by the COUNT option.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

NOTMARKED
Indicates that only those resource table records that are not marked in the result set should be considered for retrieval. The number of records that are actually retrieved is determined by the COUNT option.

You can mark resource table records by using the MARK and UNMARK commands.

POSITION*(data-value)*
Begins the retrieval process with the nth resource table record in the result set.

This value must be a number that identifies the record's relative position in the result set. The first record in a result set is identified by the number 1.

For example, to begin the retrieval process with the fifth resource table record in a result set, you would specify `POSITION(5)`.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
GETDEF
GROUP
PERFORM OBJECT.

STATUS
Retrieves only the OBJSTAT status information for the last action performed against the resource table. The records do not contain any resource table data.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the FETCH command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

NODATA
No records were found that matched the specified search criteria, for one of the following reasons:

BACKWARD	There are no more records that satisfy the search criteria in the backward direction.
FORWARD	There are no more records that satisfy the search criteria in the forward direction.

WARNING
The command completed processing with a warning, for the following reason:

AREATOOSMALL	The INTO buffer is not long enough to hold the number of records requested and available.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.
SOLRESOURCE	A required resource that is locally owned (that is, owned by the address space where the processing thread is running) is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

COUNT
FILTER
INTO
LENGTH
NOTFILTER
POSITION
RESULT
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.

NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

GET

Return a result set containing selected resource table records.

```
      GET  OBJECT(data-value)
                               COUNT(data-ref)  CRITERIA(data-area)  LENGTH(data-value)
                                                 FILTER(cpsm-token)


      PARM(data-area)  PARMLEN(data-value)NOWAIT                            CONTEXT(data-value)
                                                     TOKEN(data-value)

                               RESULT(cpsm-token)  THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-r
      SCOPE(data-value)
```

Description

This command returns a result set containing selected resource table records.

The resource table can be one that represents a CICS resource, a CICSPlex SM or CICS definition, or a CICSPlex SM run-time object.

After a GET command, the record pointer is positioned to the top of the result set (that is, the first record in the result set).

If the context and scope in effect when you issue a GET command include CICS systems that do not support the requested resource table, the request is ignored for those CICS systems.

In some CICS environments, the resource table attribute values that are returned by CICSPlex SM for:

Resource table	Attribute value	CICS Environment
CONNECT	RECOVSTATUS(NRS)	CICS/VSE
LOCTRAN	RESSEC(RESSECEXT)	CICS/VSE CICS/MVS
PROGRAM	LPASTAT(SVA)	CICS/VSE
PROGRAM	LPASTAT(NOTSVA)	CICS/VSE

do not match the CVDA values returned by CICS. The values returned by CICS conflict with CVDA values in other CICS environments. In order to retain the attributes' uniqueness, CICSPlex SM adds 9000 to the values returned by CICS.

Related commands

DISCARD, FETCH, GETDEF, QUERY, RECEIVE, REFRESH, SPECIFY FILTER, SPECIFY VIEW

Options

CONTEXT*(data-value)*
Identifies the context for this command. The context must be the 1- to 8-character name of a CMAS or CICSplex.

If you do not specify the CONTEXT option, the default context for the thread is assumed.

COUNT*(data-ref)*
Names a variable to receive the number of resource table records in the target result set after this operation is complete.

CRITERIA*(data-area)*
Identifies a buffer containing the filter expression to be used for this operation. The CRITERIA option retrieves only those resource table records that meet the specified filter criteria.

For details on how to form a filter expression, see *CICSPlex SM Application Programming Guide*.

FILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The FILTER option retrieves only those resource table records that meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

LENGTH*(data-value)*
A fullword value that specifies the length of the CRITERIA buffer.

Note: The buffer length you specify should not include any data other than a filter expression.

NOWAIT
Returns control to your program as soon as the GET command has been accepted, which allows the command to be processed asynchronously.

If you specify the NOWAIT option, you must use a subsequent RECEIVE command to test for the completion of this request. The results of an asynchronous request are returned as ASYNCREQ resource table records. For a complete description of asynchronous processing, see *CICSPlex SM Application Programming Guide*.

Note: If you specify the TOKEN option, the NOWAIT option is assumed by default.

OBJECT(*data-value*)
Identifies the resource table for which records are to be retrieved. This value must be the 1- to 8-character name of either a valid resource table or a valid view.

PARM(*data-area*)
Identifies a buffer containing the parameter expression to be used in selecting resource table records.

PARMLEN(*data-value*)
A fullword value that specifies the length of the PARM buffer.

REASON(*data-ref*)
Names a variable to receive the fullword reason value returned by this command.

RESPONSE(*data-ref*)
Names a variable to receive the fullword response value returned by this command.

RESULT(*cpsm-token*)
Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
GROUP
PERFORM OBJECT.

CICSPlex SM replaces the contents of the existing result set with the resource table records requested by this GET command. If the operation does not result in any resource table records being selected, the target result set is discarded.

If this field is:

Set to binary zero (in COBOL, C, PL/I or Assembler)
An uninitialized variable (in REXX).

CICSPlex SM creates a new result set and returns its identifying token in the same field.

SCOPE(*data-value*)
Identifies the scope for this command.

If the current context (as set by this command or a previous CONNECT or QUALIFY command) is a CICSplex and the OBJECT option identifies a CICS resource, a valid scope is required. The scope can be:

The 1- to 8-character name of the CICSplex itself
A CICS system or CICS system group within the CICSplex
A logical scope, as defined in a CICSPlex SM resource description (RESDESC).

If the current context is a CMAS or the OBJECT option identifies any other type of resource table this option has no meaning and is ignored.

If you do not specify the SCOPE option, the default scope for the thread is assumed. If the current context is a CICSplex and no default scope has been set on a CONNECT or QUALIFY command, you receive an INVALIDPARM response for the SCOPE option.

THREAD(*cpsm-token*)
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

TOKEN(*data-value*)
Defines a 1- to 4-character token that you choose to correlate an asynchronous GET request with the result of a subsequent RECEIVE command. This token is for use by your program; CICSPlex SM makes no use of the value. The token is returned by the RECEIVE command when this GET request is complete.

Conditions

The following is a list of the RESPONSE values that can be returned by the GET command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

SCHEDULED
The command has been scheduled for processing.

NODATA
No records were found that matched the specified search criteria.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
REQTIMEOUT	One of the CMASs or MASs to which the request was directed did not respond.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDATA
Invalid data was detected. The parameter that contains invalid data is returned as the reason value:

CRITERIA	An attribute value listed in the CRITERIA buffer is not valid for the specified attribute.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

CONTEXT
CRITERIA
FILTER
LENGTH
OBJECT
PARM
PARMLEN
RESULT
SCOPE
TOKEN
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CMAS	A CMAS to which the request was directed is not available.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.
MAINTPOINT	The maintenance point for the current context is not available.
SCOPE	Either none of the MASs in the specified scope are available or none of them support the requested resource table.

NOTFOUND
A not found condition occurred for the following reason:

ATTRIBUTE	An attribute specified in the CRITERIA buffer was not found for the specified resource table.

NOTPERMIT
A not permitted condition occurred for the following reason:

USRID	The user ID associated with the processing thread does not have the required security authorization.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

TABLEERROR
A resource table record is invalid for the following reason:

DATAERROR	The value associated with one or more resource table attributes is invalid. Use the FEEDBACK command to retrieve additional data about this error.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

GETDEF

Return a result set containing selected descriptive records for a resource table.

```
     GETDEF  OBJECT(data-value)                           RESOURCE(data-value)
                             ATTRIBUTE(data-value)                       COUNT(data-ref)
      REPLACE
                 RESULT(cpsm-token)  THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
      ADD
```

Description

This command returns a result set containing selected descriptive records for a resource table.

GETDEF is a variation of the GET command. GET retrieves data records for the resource represented by a table. GETDEF, on the other hand, retrieves internal data that describes the resource table itself.

The GETDEF command retrieves its data, which is called meta-data, from internal resource tables that describe each of the external resource tables These internal resource tables are called CPSM MetaData tables. The attributes of a CPSM MetaData table are the characteristics of the external table, not the resource that it represents. For a list of the CPSM MetaData resource tables that can be retrieved by GETDEF, see the description of the OBJECT option on page 38.

You can use GETDEF to find out what resource tables are available for processing by other commands. In addition, you can identify the attributes of a resource table, the values allowed for its modifiable attributes, and the actions that can be performed on it. You can also use GETDEF to request descriptions of the CPSM MetaData resource tables themselves.

You can use the GETDEF command only with resource tables supplied by CICSPlex SM. GETDEF is not valid for user-defined views of a resource table that were created by the SPECIFY VIEW command.

| You cannot use the REFRESH command to refresh the
| data records retrieved by GETDEF.

Related commands

DISCARD, FETCH, GET, LOCATE, QUERY

Options

ADD
Adds the CPSM MetaData resource table records that are being retrieved to an existing target result set. If no existing result set is specified as the target, the ADD option is ignored.

ATTRIBUTE*(data-value)*
Identifies one or more attributes of the resource table specified on the RESOURCE option for which CPSM MetaData records are to be retrieved.

Depending on which CPSM MetaData table is named in the OBJECT option, this value can be the 1- to 12-character name of a specific attribute or an asterisk (*), for all attributes in the resource table. If you do not specify the ATTRIBUTE option for an OBJECT that does not require it, data is retrieved for all attributes in the resource table.

For details on the CPSM MetaData resource tables and the valid ATTRIBUTE values for each, see the description of the OBJECT option.

COUNT*(data-ref)*
Names a variable to receive the number of resource table records in the target result set after this operation is complete.

OBJECT*(data-value)*
Identifies the type of meta-data to be retrieved for the resource table specified on the RESOURCE option. This value must be one of the following CPSM MetaData resource table names:

OBJECT
One record is returned for each instance of the resource table specified on the RESOURCE option. The record describes the resource table's general characteristics. Related options and restrictions include:

ATTRIBUTE is ignored.

RESOURCE must be a specific resource table name or * for all resource tables.

OBJACT
One record is returned for each action that is available for the resource table specified on the RESOURCE option.

Related options and restrictions include:

ATTRIBUTE is ignored.

RESOURCE must be a specific resource table name; a value of * is not allowed.

METADESC	One record is returned for each attribute of the resource table specified on the RESOURCE option. Each record provides only the basic structure of the attribute, including the name, data type, length, and offset in the resource table. Such information might be useful for accessing the attribute fields in a buffer returned by the FETCH command. Related options and restrictions include: ATTRIBUTE can be a specific attribute name or * for all attributes in the resource table. RESOURCE must be a specific resource table name; a value of * is not allowed.
ATTR	One record is returned for each attribute of the resource table specified on the RESOURCE option. Each record provides complete information about the attribute. Related options and restrictions include: ATTRIBUTE can be a specific attribute name or * for all attributes in the resource table. RESOURCE must be a specific resource table name; a value of * is not allowed.
ATTRAVA	One record is returned for each of the EYUDA or CVDA values that are valid for the specified attribute. Related options and restrictions include: ATTRIBUTE must be the name of a specific attribute that has a data type of EYUDA, CVDAS, or CVDAT. RESOURCE must be a specific resource table name; a value of * is not allowed. **Note:** The AVAAVAIL attribute of the ATTR internal resource table indicates whether an AVA list is available for a given attribute.

REASON(*data-ref*)
Names a variable to receive the fullword reason value returned by this command.

REPLACE
Deletes the contents of an existing target result set and replaces them with the results of this operation. If the operation does not result in any CPSM MetaData resource table records being selected, the target result set is discarded.

If no existing result set is specified as the target, the REPLACE option is ignored.

RESOURCE(*data-value*)
Identifies the resource table for which CPSM MetaData records are to be retrieved.

If you specify the ATTRIBUTE option, this value must be the 1- to 8-character name of a specific CICSPlex SM resource table. Otherwise, you can specify a value of asterisk (*) to retrieve data for all resource tables.

Note: You can use GETDEF only with resource tables supplied by CICSPlex SM. GETDEF is not valid for user-defined views of a resource table that were created by the SPECIFY VIEW command.

RESPONSE(*data-ref*)
Names a variable to receive the fullword response value returned by this command.

RESULT(*cpsm-token*)
Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
GROUP
PERFORM OBJECT.

CICSPlex SM replaces the contents of the existing result set with the resource table records requested by this GETDEF command. If the operation does not result in any resource table records being selected, the target result set is discarded.

If this field is:

Set to binary zero (in COBOL, C, PL/I or Assembler)
An uninitialized variable (in REXX).

CICSPlex SM creates a new result set and returns its identifying token in the same field.

THREAD(*cpsm-token*)
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the GETDEF command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

NODATA
OBJACT was specified with the OBJECT option, but there are no actions defined for the specified RESOURCE.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INCOMPATIBLE
An incompatible condition occurred for the following reason:

INVALIDOBJ	The target result set specified on the RESULT option is not compatible with the output of this command. The result set must contain the same type of meta-data (as specified on the OBJECT option) as the command produces.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

ATTRIBUTE
OBJECT
RESOURCE
RESULT
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

TABLEERROR
A resource table record is invalid for one of the following reasons:

DATAERROR	The value associated with one or more resource table attributes is invalid. Use the FEEDBACK command to retrieve additional data about this error.
INVALIDVER	The specified version of the resource table is not supported by CICSPlex SM.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

GROUP

Return a summarized result set.

```
                                                   ALL
     GROUP  BY(data-value)  FROM(cpsm-token)  TO(cpsm-token)
                                                   FILTER(cpsm-token)
                                                   MARKED
                                                   NOTFILTER(cpsm-token)
                                                   NOTMARKED

                                                                 THREAD(cpsm-token)
      COUNT(data-ref)  SUMOPT(data-area)  LENGTH(data-value)

    RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command returns a summarized result set by grouping some or all of the resource table records in a result set.

The target result set can be an existing result set or a new one that is created by this process. If you specify an existing result set as the target of a GROUP command:

- It must be a summarized result set that was produced by a previous GROUP command against the same source result set.
- It must contain the same type of resource table records currently found in the source result set.
- The existing records in the result set are overwritten.

To create a summarized result set from selected records of a source result set, you can use:

- The SPECIFY FILTER command to define a filter for the source result set.
- The MARK and UNMARK commands to mark records in the source result set.

For more information on processing summarized result sets, see *CICSPlex SM Application Programming Guide*.

Related commands

DISCARD, FETCH, GET, LOCATE, MARK, ORDER, QUERY, SPECIFY FILTER

Options

ALL
Summarizes all the resource table records in the source result set.

BY*(data-value)*
Identifies the resource table attribute whose value is to be used as the grouping factor for this operation. This value must be the 1- to 12-character name of a valid attribute for the resource table.

COUNT*(data-ref)*
Names a variable to receive the number of resource table records in the target result set after this operation is complete.

FILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The FILTER option summarizes only those resource table records that meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

FROM*(cpsm-token)*
Identifies the source result set for this operation. The result set can be one produced by any of these commands:

COPY
GET
PERFORM OBJECT.

Note: If you discard the source result set, all of the summarized result sets that were created from it are also discarded.

LENGTH*(data-value)*
A fullword value that specifies the length of the SUMOPT buffer.

Note: The buffer length you specify should not include any data other than a summary expression.

MARKED
Summarizes only those resource table records that are marked in the source result set. You can mark resource table records by using the MARK and UNMARK commands.

NOTFILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The NOTFILTER option summarizes only those resource table records that do not meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

NOTMARKED
Summarizes only those resource table records that are not marked in the source result set. You can mark resource table records by using the MARK and UNMARK commands.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

SUMOPT*(data-area)*
Identifies a buffer containing the summary expression to be used for this operation. The SUMOPT value overrides the default summary options for the resource table attributes.

For details on how to form a summary expression, see *CICSPlex SM Application Programming Guide*. For a list of the default summary options for a given resource table, see the *CICSPlex SM Resource Tables Reference*.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

TO*(cpsm-token)*
Identifies the target result set for this operation.

If this field is:

Set to binary zero (in COBOL, C, PL/I or Assembler)
An uninitialized variable (in REXX).

CICSPlex SM creates a new summarized result set and returns its identifying token in the same field.

Otherwise, you can specify an existing summarized result that was produced by a previous GROUP command against the result set specified in the FROM option. That is, you can reuse a summarized result set, but only to resummarize the records in the same result set.

Note: If you specify the token of a previously produced summarized result set, make sure the result set still exists. When you discard a source result set, all of the summarized result sets that were created from it are also discarded.

Conditions

The following is a list of the RESPONSE values that can be returned by the GROUP command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

NODATA
No records were found that matched the specified search criteria.

BUSY
A busy condition occurred for one of the following reasons:

FROM	The source result set specified on the FROM option is being processed by another command.
TO	The target result set specified on the TO option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
| **SOCRESOURCE**	A required resource that is owned by the CMAS is not available. |

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

BY
FILTER
FROM
LENGTH
NOTFILTER
SUMOPT
THREAD
TO.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

LISTEN

Request a notification be sent to the processing thread.

```
     LISTEN  EVENT(data-value)  NOTIFICATION(data-ref)
                                                FILTER(cpsm-token)       TOKEN(data-value)
                                                NOTFILTER(cpsm-token)

                             THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
      CONTEXT(data-value)
```

Description

This command requests that a notification be sent to the processing thread when a specific event occurs in the CICSplex.

An event is represented by a resource table with a type of CPSM Notification.

The LISTEN command is used in conjunction with the RECEIVE command. If you use LISTEN to request notification of an event, you must use a subsequent RECEIVE command to retrieve information about the event.

An API processing thread can have a maximum of 256 completed asynchronous requests outstanding at one time. If you do not issue the RECEIVE command at regular intervals and your processing thread reaches its maximum of 256, asynchronous requests are discarded and are not processed. For a complete description of asynchronous processing, see *CICSPlex SM Application Programming Guide*.

Related commands

ADDRESS, CANCEL, RECEIVE

Options

CONTEXT(*data-value*)
Identifies the context for this command. The context must be the 1- to 8-character name of a CMAS or CICSplex.

If you do not specify the CONTEXT option, the default context for the thread is assumed.

EVENT(*data-value*)
Identifies the resource table that represents the event to be listened for. This value must be the 1- to 8-character name of a valid CPSM Notification resource table. For a list of the CICSPlex SM resource tables by type, see *CICSPlex SM Application Programming Guide*.

FILTER(*cpsm-token*)
Identifies a filter to be used for this operation. The FILTER option listens for only those events that meet the specified filter criteria.

Using the FILTER option, you can limit the notifications you receive to events that are associated with a specific CMAS or CICSplex. For example, you could create a filter like this:

```
PLEXNAME=EYUPLX 1.
```

and specify that filter on the LISTEN command to be notified only of events generated by CICSplex EYUPLX01.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

NOTFILTER(*cpsm-token*)
Identifies a filter to be used for this operation. The NOTFILTER option listens for only those events that do not meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

NOTIFICATION(*data-ref*)
Names a variable to receive the fullword token that CICSPlex SM assigns to this notification request.

This identifying token must be specified on the CANCEL command when you want to cancel the notification request.

REASON(*data-ref*)
Names a variable to receive the fullword reason value returned by this command.

RESPONSE(*data-ref*)
Names a variable to receive the fullword response value returned by this command.

THREAD(*cpsm-token*)
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

TOKEN(*data-value*)
Defines a 1- to 4-character token that you choose to correlate this LISTEN request with the result of a subsequent RECEIVE command. This token is for use by your program; CICSPlex SM makes no use of the value. The token is returned by the RECEIVE command when an event of the specified type occurs.

Conditions

The following is a list of the RESPONSE values that can be returned by the LISTEN command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INCOMPATIBLE
An incompatible condition occurred for the following reason:

INVALIDEVT	The specified event is not compatible with the filter specified on the FILTER or NOTFILTER option.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

CONTEXT
EVENT
FILTER
NOTFILTER
NOTIFICATION
THREAD
TOKEN.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread was trying to connect is not available for API processing.
PLEXMGR	The CMAS to which the processing thread is currently connected does not participate in managing the specified CICSplex and no other CMAS is available that does manage the CICSplex.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

LOCATE

Position the record pointer within a result set.

```
              TOP
      LOCATE                                        RESULT(cpsm-token)  THREAD(cpsm-token)
              BOTTOM
              POSITION(data-value)
              FORWARD(data-value)
              BACKWARD(data-value)
                                     FIRST
              FILTER(cpsm-token)
              MARKED                     LAST
              NOTFILTER(cpsm-token)      NEXT
              NOTMARKED                  PREV
     RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command positions the record pointer within a result set.

API commands that manipulate records or update the data in a result set affect the position of the record pointer:

– After a GET command, the pointer is positioned to the top of the result set.

– After a FETCH command, the pointer is positioned to the next record in the result set (that is, the record following the last record fetched in whichever direction the pointer was moving, forward or backward). If no records were fetched (because no records matched the specified criteria), the pointer is positioned to the top or bottom of the result set, depending on which direction it was moving.

After issuing any other command that manipulates records or updates data, the position of the record pointer depends on a combination of factors, including the options that you specified on the command. To be certain of the pointer's location, you should use the LOCATE command to explicitly position it within the result set.

The LOCATE command skips over any deleted records in the result set. If you try to position the record pointer to a deleted record, you receive a RESPONSE value of NODATA.

Related commands

COPY, DELETE, FETCH, GETDEF, GROUP, MARK, ORDER, PERFORM OBJECT, PERFORM SET, REFRESH, SET, SPECIFY FILTER, UNMARK

Options

BACKWARD*(data-value)*
Moves the record pointer backward by the specified number of resource table records.

If the pointer reaches the top of the result set, it remains positioned on the first resource table record. The pointer does not continue moving backward to the bottom of the result set.

BOTTOM
Moves the record pointer to the the last resource table record in the result set.

FILTER*(cpsm-token)*
Identifies a filter to be used for this operation.

The FILTER option positions the record pointer to a resource table record that meets the specified filter criteria. The FIRST, LAST, NEXT, or PREV option determines where in the result set the search begins and in what direction it continues.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

FIRST
Begins a search based upon filter or marking criteria with the first resource table record in the result set. The search continues in a forward direction through the result set until a match is found.

FORWARD*(data-value)*
Moves the record pointer forward by the specified number of resource table records.

If the pointer reaches the bottom of the result set, it remains positioned on the last resource table record. The pointer does not continue moving forward to the top of the result set.

LAST
Begins a search based upon filter or marking criteria with the last resource table record in the result set. The search continues in a backward direction through the result set until a match is found.

MARKED
Positions the record pointer to a resource table record that is marked. The FIRST, LAST, NEXT, or PREV option determines where in the result set the search begins and in what direction it continues.

You can mark resource table records by using the MARK and UNMARK commands.

NEXT
Begins a search based upon filter or marking criteria with the current resource table record in the result set. The search continues in a forward direction through the result set until a match is found.

NOTFILTER(*cpsm-token)*
Identifies a filter to be used for this operation.

The NOTFILTER option positions the record pointer to a resource table record that does not meet the specified filter criteria. The FIRST, LAST, NEXT, or PREV option determines where in the result set the search begins and in what direction it continues.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

NOTMARKED
Positions the record pointer to a resource table record that is not marked. The FIRST, LAST, NEXT, or PREV option determines where in the result set the search begins and in what direction it continues.

You can mark resource table records by using the MARK and UNMARK commands.

POSITION(*data-value)*
Moves the record pointer to the nth resource table record in the result set.

This value must be a number that identifies the record's relative position in the result set. The first record in a result set is identified by the number 1.

For example, to move the record pointer to the fifth resource table record in a result set, you would specify `POSITION(5)`.

PREV
Begins a search based upon filter or marking criteria with the previous resource table record in the result set. The search continues in a backward direction through the result set until a match is found.

REASON(*data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE(*data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT(*cpsm-token)*
Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
GETDEF
GROUP
PERFORM OBJECT.

THREAD(*cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

TOP
Moves the record pointer to the first resource table record in the result set.

Conditions

The following is a list of the RESPONSE values that can be returned by the LOCATE command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

NODATA
No records were found that matched the specified search criteria, for one of the following reasons:

BACKWARD	There are no more records that satisfy the search criteria in the backward direction.
FORWARD	There are no more records that satisfy the search criteria in the forward direction.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.
SOLRESOURCE	A required resource that is locally owned (that is, owned by the address space where the processing thread is running) is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.

EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

BACKWARD
FILTER
FORWARD
NOTFILTER
POSITION
RESULT
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread was trying to connect is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

MARK

Mark selected resource table records in a result set.

```
           CURRENT
    MARK
           ALL                                      COUNT(data-ref)
            FILTER(cpsm-token)
           NOTFILTER(cpsm-token)
           POSITION(data-value)
          PARM(data-area)  PARMLEN(data-value)
                                                       RESULT(cpsm-token)  THREAD(cpsm-token)
     INTO(data-area)  LENGTH(data-ref)      RESET
   RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command marks selected resource table records in a result set.

The MARK command always begins processing with the last record that was fetched, rather than the next one in the result set.

Any resource table records that you marked in the result set previously remain marked unless you use the RESET option.

Related commands

COPY, DELETE, FETCH, GROUP, LOCATE, PERFORM SET, REFRESH, SET, SPECIFY FILTER, UNMARK

Options

ALL
Marks all the resource table records in the result set. When you specify ALL, the RESET option is ignored.

COUNT*(data-ref)*
Names a variable to receive the number of resource table records that could not be marked.

CURRENT
Marks only the current resource table record.

FILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The FILTER option marks only those resource table records that meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

INTO*(data-area)*
Identifies a buffer to receive a list of resource table records that could not be marked.

This buffer must be long enough to hold the maximum number of record numbers that could result from your MARK request (in the event that none of them can be marked). Record numbers are listed individually (not by range) in the INTO buffer and are separated by commas.

Note: If you receive a RESPONSE value of WARNING AREATOOSMALL (because the buffer was not long enough), the data returned in this buffer represents a partial list of the records that could not be marked.

LENGTH*(data-ref)*
A fullword value that specifies the length of the INTO buffer.

The value that CICSPlex SM returns in this field depends on the RESPONSE value for the MARK command:

OK The actual length of the data returned in the INTO buffer.

WARNING AREATOOSMALL
The buffer length that would be required to hold a complete list of records that could not be marked.

NOTFILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The NOTFILTER option marks only those resource table records that do not meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

PARM*(data-area)*
Identifies a buffer containing the parameter expression that lists the resource table records to be marked.

The parameter expression for the MARK command is a character string of record numbers. For example:

```
PARM('1,3,6:9,24.')
```

To specify individual records, separate the record numbers with a comma. To specify a range of records, separate the low and high record numbers with a colon. The whole parameter expression must end with a period.

For details on how to use a parameter expression with the MARK command, see *CICSPlex SM Application Programming Guide*.

PARMLEN*(data-value)*
A fullword value that specifies the length of the PARM buffer.

POSITION*(data-value)*
Marks the nth resource table record in the result set.

This value must be a number that identifies the record's relative position in the result set. The first record in a result set is identified by the number 1.

For example, to mark the fifth resource table record in a result set, you would specify `POSITION(5)`.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESET
Removes any marks previously placed on resource table records in the result set and marks only those records you identify in the current MARK request.

If you do not use the RESET option, any records that you marked previously remain marked. That is, the records identified in the current MARK request are marked in addition to any previously marked records.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
GETDEF
GROUP
PERFORM OBJECT.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the MARK command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

NODATA
No records were found that matched the specified search criteria.

WARNING
The command completed processing with a warning, for one of the following reasons:

AREATOOSMALL	You specified the INTO and LENGTH options, but the buffer was not long enough to hold the string of records that could not be marked.
DATAERROR	One or more of the records specified in the PARM buffer could not be found to be marked. If you specified the COUNT option, the number of records that could not be marked is returned. If you specified the INTO and LENGTH options, a list of the records is returned in the buffer.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.
SOLRESOURCE	A required resource that is locally owned (that is, owned by the address space where the | processing thread is running) is not | available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

COUNT
FILTER
INTO
LENGTH
NOTFILTER
PARM
PARMLEN
RESULT
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

ORDER

Sort the resource table records in a result set.

```
      ORDER  BY(data-area)  LENGTH(data-value)  RESULT(cpsm-token)  THREAD(cpsm-token)  RESPONSE(data-ref)
     REASON(data-ref)
```

Description

This command sorts the resource table records in a result set into a user-specified order.

By default, records are sorted by the key attributes for the resource table.

The sort order you specify for a result set remains in effect until you issue another ORDER command.

If the result set contains deleted records, those records are included in the sorting process. They are sorted by the same attributes as other records and their position in the newly ordered result set may be difficult to determine. To prevent this happening, issue the REFRESH command before issuing ORDER; REFRESH removes any deleted records from the result set.

Related commands

COPY, GET, GETDEF, GROUP, LOCATE, PERFORM OBJECT

Options

BY*(data-area)*
: Identifies a buffer containing the order expression to be used for this operation.

An order expression is a list of attributes to be used in sorting the resource table records. For example:

```
CICSSYS,TRANID.
```

where the attribute names are separated by commas or blank spaces and the whole expression ends with a period.

In this example, the resource table records are sorted using CICS system name as the primary sort key and transaction ID as the secondary key. The default sort order is ascending. To sort attribute values in descending order, add /D to the end of the attribute name.

For more information on using order expressions with the ORDER command, see *CICSPlex SM Application Programming Guide*.

Note: You cannot specify the EYU_CICSNAME or EYU_CICSREL attributes in an order expression.

LENGTH*(data-value)*
: A fullword value that specifies the length of the BY buffer.

Note: The buffer length you specify should not include any data other than an order expression.

REASON*(data-ref)*
: Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
: Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
: Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
GETDEF
GROUP
PERFORM OBJECT.

THREAD*(cpsm-token)*
: Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the ORDER command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
: The command completed processing successfully.

BUSY
: A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
: An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.

NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
| **SOCRESOURCE**	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

BY
LENGTH
RESULT
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

PERFORM OBJECT

Perform an action on one or more resources.

```
     PERFORM  OBJECT(data-value)  ACTION(data-value)
                                                      COUNT(data-ref)


     CRITERIA(data-area)  LENGTH(data-value)       NOREFRESH NOWAIT
      FILTER(cpsm-token)                                                 TOKEN(data-value)


     PARM(data-area)  PARMLEN(data-value)       CONTEXT(data-value)      SCOPE(data-value)
    RESULT(cpsm-token)  THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command performs an action on one or more resources.

The resources to be acted upon by PERFORM OBJECT do not have to exist as records in a result set; a result set is implicitly created by this process.

If the context and scope in effect when you issue a PERFORM OBJECT command include CICS systems that do not support the requested action, the request is ignored for those CICS systems.

Related commands

DISCARD, GET, LOCATE, PERFORM SET, QUERY, SET, SPECIFY FILTER

Options

ACTION(*data-value*)
Identifies the action to be performed. This value must be the 1- to 12-character name of a valid action for the resource table.

For a description of the actions that are valid for a given resource table, see the *CICSPlex SM Resource Tables Reference*.

CONTEXT(*data-value*)
Identifies the context for this command. The context must be the 1- to 8-character name of a CMAS or CICSplex.

If you do not specify the CONTEXT option, the default context for the thread is assumed.

COUNT(*data-ref*)
Names a variable to receive the number of resource table records in the target result set after this operation is complete.

CRITERIA(*data-area*)
Identifies a buffer containing the filter expression to be used for this operation. The CRITERIA option retrieves only those resource table records that meet the specified filter criteria.

For details on how to form a filter expression, see *CICSPlex SM Application Programming Guide*.

FILTER(*cpsm-token*)
Identifies a filter to be used for this operation. The FILTER option retrieves only those resource table records that meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

LENGTH(*data-value*)
A fullword value that specifies the length of the CRITERIA buffer.

Note: The buffer length you specify should not include any data other than a filter expression.

NOREFRESH
Specifies that the resource table records in the result set created by PERFORM OBJECT should not be refreshed. The records reflect the status of the resources when the result set was created.

If you do not specify the NOREFRESH option, the resource table records are refreshed to reflect the resource status after this operation is complete.

NOWAIT
Returns control to your program as soon as the PERFORM OBJECT command has been accepted, which allows the command to be processed asynchronously.

If you specify the NOWAIT option, you must use a subsequent RECEIVE command to test for the completion of this request. The results of an asynchronous request are returned as ASYNCREQ resource table records. For a complete description of asynchronous processing, see *CICSPlex SM Application Programming Guide*.

Note: If you specify the TOKEN option, the NOWAIT option is assumed by default.

OBJECT(*data-value*)
Identifies the resource table against which the action is to be performed. This value must be the 8-character name of a valid resource table.

PARM(*data-area*)
: Identifies a buffer containing the parameter expression to be used in performing the action.

  For details on how to use a parameter expression with the PERFORM OBJECT command, see *CICSPlex SM Application Programming Guide*. For a description of the parameters that are required for a given resource table action, see the *CICSPlex SM Resource Tables Reference*.

PARMLEN(*data-value*)
: A fullword value that specifies the length of the PARM buffer.

REASON(*data-ref*)
: Names a variable to receive the fullword reason value returned by this command.

RESPONSE(*data-ref*)
: Names a variable to receive the fullword response value returned by this command.

RESULT(*cpsm-token*)
: Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

  COPY
  GET
  GROUP
  PERFORM OBJECT.

  CICSPlex SM replaces the contents of the existing result set with the resource table records requested by this PERFORM OBJECT command.

  If this field is:

  Set to binary zero (in COBOL, C, PL/I or Assembler)
  An uninitialized variable (in REXX).

  CICSPlex SM creates a new result set and returns its identifying token in the same field.

SCOPE(*data-value*)
: Identifies the scope for this command.

  To use the SCOPE option, the current context (as set by this command or a previous CONNECT or QUALIFY command) must be a CICSplex. The scope can be:

  The 1- to 8-character name of the CICSplex itself
  A CICS system or CICS system group within the CICSplex
  A logical scope, as defined in a CICSPlex SM resource description (RESDESC).

  If the current context is a CMAS, this option has no meaning and is ignored.

  If you do not specify the SCOPE option, the default scope for the thread is assumed.

THREAD(*cpsm-token*)
: Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

TOKEN(*data-value*)
: Defines a 1- to 4-character token that you choose to correlate an asynchronous PERFORM OBJECT request with the result of a subsequent RECEIVE command. This token is for use by your program; CICSPlex SM makes no use of the value. The token is returned by the RECEIVE command when this PERFORM OBJECT request is complete.

Conditions

The following is a list of the RESPONSE values that can be returned by the PERFORM OBJECT command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
: The command completed processing successfully.

SCHEDULED
: The command has been scheduled for processing.

NODATA
: No records were found that matched the specified search criteria.

BUSY
: A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
: An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
REQTIMEOUT	One of the CMASs or MASs to which the request was directed did not respond.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
: The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDATA
: Invalid data was detected. The parameter that contains invalid data is returned as the reason value:

PARM	An attribute value listed in the PARM buffer is not valid for the specified attribute.
CRITERIA	An attribute value listed in the CRITERIA buffer is not valid for the specified attribute.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

ACTION
CONTEXT
CRITERIA
FILTER
LENGTH
OBJECT
PARM
PARMLEN
RESULT
SCOPE
THREAD
TOKEN.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CMAS	A CMAS to which the request was directed is not available.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.
MAINTPOINT	The maintenance point for the current context is not available.
PLEXMGR	The CMAS to which the processing thread is currently connected does not participate in managing the specified CICSplex and no other CMAS is available that does manage the CICSplex.
SCOPE	Either none of the MASs in the specified scope are available or none of them support the requested action.

NOTFOUND
A not found condition occurred for one of the following reasons:

ACTION	The action specified on the ACTION option was not found for the specified resource table.
ATTRIBUTE	An attribute specified in the CRITERIA or PARM buffer was not found for the specified resource table.

NOTPERMIT
A not permitted condition occurred for the following reason:

USRID	The user ID associated with the processing thread does not have the required security authorization.

TABLEERROR
A resource table record is invalid for the following reason:

DATAERROR	The value associated with one or more resource table attributes is invalid. This error can occur if the resource table is missing required fields, contains one or more conflicting fields, or is a duplicate. | For BAS this error can also occur if | you do not have the required | security authorization. Use the FEEDBACK command to retrieve additional data about this error.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

PERFORM SET

Performs an action on one or more resources.

```
                                ALL
      PERFORM  SET  ACTION(data-value)
                                CURRENT                    NOREFRESH
                                FILTER(cpsm-token)
                                MARKED
                                NOTFILTER(cpsm-token)
                                NOTMARKED
                                POSITION(data-value)
                                                                                RESULT(cpsm-token)
      NOWAIT                    PARM(data-area)  PARMLEN(data-value)
             TOKEN(data-value)

     THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command performs an action on one or more resources as represented by resource table records in an existing result set. If the context and scope in effect when you issue a PERFORM SET command include CICS systems that do not support the requested action, the request is ignored for those CICS systems.

Related commands

LOCATE, MARK, PERFORM OBJECT, SET, SPECIFY FILTER

Options

ACTION*(data-value)*
: Identifies the action to be performed. This value must be the 1- to 12-character name of a valid action for the resource table.

For a description of the actions that are valid for a given resource table, see the *CICSPlex SM Resource Tables Reference*.

ALL
: Performs the specified action against all the resource table records in the result set.

CURRENT
: Performs the specified action against only the current resource table record.

FILTER*(cpsm-token)*
: Identifies a filter to be used for this operation. The FILTER option performs the action against only those resource table records that meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

MARKED
: Performs the specified action against only those resource table records that are marked in the result set. You can mark resource table records by using the MARK and UNMARK commands.

NOREFRESH
: Specifies that the resource table records in the source result set should not be refreshed. The records reflect the status of the resources before the PERFORM SET command was processed.

If you do not specify the NOREFRESH option, the resource table records are refreshed to reflect the resource status after this operation is complete.

NOTFILTER*(cpsm-token)*
: Identifies a filter to be used for this operation. The NOTFILTER option performs the action against only those resource table records that do not meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

NOTMARKED
: Performs the specified action against only those resource table records that are not marked in the result set. You can mark resource table records by using the MARK and UNMARK commands.

NOWAIT
: Returns control to your program as soon as the PERFORM SET command has been accepted, which allows the command to be processed asynchronously.

If you specify the NOWAIT option, you must use a subsequent RECEIVE command to test for the completion of this request. The results of an asynchronous request are returned as ASYNCREQ resource table records. For a complete description of asynchronous processing, see *CICSPlex SM Application Programming Guide*.

Note: If you specify the TOKEN option, the NOWAIT option is assumed by default.

PARM*(data-area)*
Identifies a buffer containing the parameter expression to be used in performing the action.

For details on how to use a parameter expression with the PERFORM SET command, see *CICSPlex SM Application Programming Guide*. For a description of the parameters that are required for a given resource table action, see the *CICSPlex SM Resource Tables Reference*.

PARMLEN*(data-value)*
A fullword value that specifies the length of the PARM buffer.

POSITION*(data-value)*
Performs the specified action against the nth resource table record in the result set.

This value must be a number that identifies the record's relative position in the result set. The first record in a result set is identified by the number 1.

For example, to perform the specified action on the fifth resource table record in a result set, you would specify `POSITION(5)`.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
GROUP
PERFORM OBJECT.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

TOKEN*(data-value)*
Defines a 1- to 4-character token that you choose to correlate an asynchronous PERFORM SET request with the result of a subsequent RECEIVE command. This token is for use by your program; CICSPlex SM makes no use of the value. The token is returned by the RECEIVE command when this PERFORM SET request is complete.

Conditions

The following is a list of the RESPONSE values that can be returned by the PERFORM SET command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

SCHEDULED
The command has been scheduled for processing.

NODATA
No records were found that matched the specified search criteria. If the ALL option was specified, the following reason may be returned:

FORWARD	There are no more records that satisfy the search criteria in the forward direction.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
| **REQTIMEOUT**	One of the CMASs or MASs to which the request was directed did not respond.
| **SOCRESOURCE**	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDATA
Invalid data was detected. The parameter that contains invalid data is returned as the reason value:

PARM	An attribute value listed in the PARM buffer is not valid for the specified attribute.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

ACTION
FILTER
NOTFILTER
PARM

PARMLEN
POSITION
RESULT
THREAD
TOKEN.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CMAS	A CMAS to which the request was directed is not available.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.
MAINTPOINT	The maintenance point for the current context is not available.
PLEXMGR	The CMAS to which the processing thread is currently connected does not participate in managing the specified CICSplex and no other CMAS is available that does manage the CICSplex.
SCOPE	Either none of the MASs in the specified scope are available or none of them support the requested action.

NOTFOUND
A not found condition occurred for one of the following reasons:

ACTION	The action specified on the ACTION option was not found for the specified resource table.
ATTRIBUTE	An attribute specified in the CRITERIA or PARM buffer was not found for the specified resource table.

NOTPERMIT
A not permitted condition occurred for the following reason:

USRID	The user ID associated with the processing thread does not have the required security authorization.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

TABLEERROR
A resource table record is invalid for the following reason:

DATAERROR	The value associated with one or more resource table attributes is invalid. This error can occur if the resource table is missing required fields, contains one or more conflicting fields, or is a duplicate. | For BAS this error can also occur if | you do not have the required | security authorization. Use the FEEDBACK command to retrieve additional data about this error.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

QUALIFY

Define the CICSPlex SM context and scope.

```
      QUALIFY  CONTEXT(data-value)                        THREAD(cpsm-token)  RESPONSE(data-ref)
                                 SCOPE(data-value)
     REASON(data-ref)
```

Related commands

CONNECT

Description

This command defines the CICSPlex SM context and scope for subsequent commands issued by an API processing thread.

Options

CONTEXT*(data-value)*
: Identifies the context for subsequent commands issued against this thread. The context must be the 1- to 8-character name of a CMAS or CICSplex.

  The specified context remains in effect for the thread until you override it or change it on a subsequent command.

REASON*(data-ref)*
: Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
: Names a variable to receive the fullword response value returned by this command.

SCOPE*(data-value)*
: Identifies the scope for subsequent commands issued against this thread.

  The SCOPE option qualifies the CONTEXT option. When the context is a CICSplex, the scope can be:

  - The 1- to 8-character name of the CICSplex itself
  - A CICS system or CICS system group within the CICSplex
  - A logical scope, as defined in a CICSPlex SM resource description (RESDESC).

  When the context is a CMAS, this option has no meaning and is ignored.

  The specified scope remains in effect for the thread unless you override it for a specific command or change it by issuing another QUALIFY command. If you do not specify the SCOPE option, no scope value is assumed (that is, the default scope established for the thread by the CONNECT command is not retained).

  Note: Certain API commands require a valid scope when the context is a CICSplex. If you do not specify a scope on the QUALIFY command, then you must specify the SCOPE option when you issue any of these commands for a resource table that represents a CICS resource:

  GET
  PERFORM OBJECT
  PERFORM SET
  REFRESH
  SET.

THREAD*(cpsm-token)*
: Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the QUALIFY command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
: The command completed processing successfully.

ENVIRONERROR
: An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
: The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
: An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

  CONTEXT
  SCOPE

THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.
PLEXMGR	The CMAS to which the processing thread is currently connected does not participate in managing the specified CICSplex and no other CMAS is available that does manage the CICSplex.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

QUERY

Retrieve information about a result set and the resource table records it contains.

```
     QUERY  RESULT(cpsm-token)
                              COUNT(data-ref)      DATALENGTH(data-ref)      OBJECT(data-ref)
                                                THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
      TYPE(data-ref)      CONTEXT(data-ref)
```

Description

This command retrieves information about a result set and the resource table records it contains.

You can use the QUERY command to determine:

- The context and scope of the result set
- The type of resource table records the result set contains
- Whether the records are from the CICSPlex SM resource table or a user-defined view of that table
- The number of resource table records in the result set
- The length of the resource table records

For programs written in REXX, issuing the QUERY command is the only way to determine the length of a given resource table record.

Related commands

COPY, GET, GETDEF, GROUP, PERFORM OBJECT

Options

CONTEXT*(data-ref)*
Names a variable to receive the context associated with the result set.

COUNT*(data-ref)*
Names a variable to receive the number of resource table records in the result set.

DATALENGTH*(data-ref)*
Names a variable to receive the length of the resource table records in the result set.

OBJECT*(data-ref)*
Names a variable to receive the name of the resource table currently associated with the result set.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
GETDEF
GROUP
PERFORM OBJECT.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

TYPE*(data-ref)*
Names a variable to receive a 1-character value that indicates what type of records are in the result set:

T	Resource tables supplied by CICSPlex SM.
V	Views of a resource table created by a SPECIFY VIEW command issued previously on this processing thread.

Conditions

The following is a list of the RESPONSE values that can be returned by the QUERY command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

CONTEXT
DATALENGTH
OBJECT
RESULT
THREAD
TYPE.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

RECEIVE

Receive the output from completed asynchronous requests.

```
      RECEIVE  INTO(data-area)  LENGTH(data-ref)  OBJECT(data-ref)
                                                           TOKEN(data-ref)

       WAIT
                         THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
       DELAY(data-value)
       IMMEDIATE
```

Description

This command receives the output from completed asynchronous requests associated with the processing thread.

Asynchronous output can result if you previously issued either a LISTEN command or one of these commands with the NOWAIT option:

- GET
- PERFORM OBJECT
- PERFORM SET
- REFRESH
- SET.

To determine if there is any asynchronous output to be received, issue the ADDRESS command and check the SENTINEL value before you issue the RECEIVE command.

An API processing thread can have a maximum of 256 completed asynchronous requests outstanding at one time. If you do not issue the RECEIVE command at regular intervals and your processing thread reaches its maximum of 256, asynchronous requests are discarded and are not processed. For a complete description of asynchronous processing, see *CICSPlex SM Application Programming Guide*.

Related commands

ADDRESS, GET, LISTEN, PERFORM OBJECT, PERFORM SET, REFRESH, SET

Options

DELAY(*data-value*)
| Specifies the number of seconds that processing will
| wait if no output is available when the RECEIVE
| command is issued. At the end of the specified number
| of seconds, control returns to the processing thread,
| whether or not any output becomes available. If output
| becomes available during the delay period, control
| returns to the processing thread. If output is immediately available, there is no delay; control returns immediately to the processing thread.

Note: If you specify a value of 0, control returns to the processing thread immediately. However, the DELAY option incurs additional API processing overhead, so it is recommended that you use the IMMEDIATE option instead.

IMMEDIATE
Returns control to the processing thread immediately, whether or not any output is available.

INTO(*data-area*)
Identifies a buffer to receive asynchronous output, if any is available for this thread. This buffer must be long enough to hold all the output being received.

The output returned can be:

A resource table record representing an event named in a previous LISTEN command

An ASYNCREQ resource table record representing completion of an asynchronous GET, PERFORM, REFRESH, or SET request.

LENGTH(*data-ref*)
A fullword value that specifies the length of the INTO buffer.

After the operation is complete, this field is set to the actual length of the data returned in the INTO buffer. If the operation cannot complete because the buffer is not long enough, this field is set to the length that is required.

OBJECT(*data-ref*)
Names a variable to receive a resource table name, if output is available for this thread.

REASON(*data-ref*)
Names a variable to receive the fullword reason value returned by this command.

RESPONSE(*data-ref*)
Names a variable to receive the fullword response value returned by this command.

THREAD(*cpsm-token*)
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

WAIT
Waits until asynchronous output becomes available before returning control to the processing thread.

Note: The WAIT option waits indefinitely for asynchronous output. Be sure to verify that there are completed asynchronous requests outstanding by issuing the ADDRESS command before you issue RECEIVE.

TOKEN*(data-ref)*
Names a variable to receive the user-defined token associated with the asynchronous output. This value is the token you defined on the GET, LISTEN, PERFORM, REFRESH or SET command that produced the output.

Conditions

The following is a list of the RESPONSE values that can be returned by the RECEIVE command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

NODATA
There was no data to receive.

WARNING
The command completed processing with a warning, for the following reason:

AREATOOSMALL	The INTO buffer is not long enough to hold the number of records requested and available.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

DELAY
INTO
LENGTH
OBJECT
THREAD
TOKEN.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

REFRESH

Refreshes the data for resource table records.

```
    REFRESH
             ALL                                       FORWARD
             CURRENT               COUNT(data-value)
             FILTER(cpsm-token)                        BACKWARD
             MARKED
             NOTFILTER(cpsm-token)
             NOTMARKED
                                      RESULT(cpsm-token)  THREAD(cpsm-token)  RESPONSE(data-ref)
     NOWAIT
             TOKEN(data-value)
   REASON(data-ref)
```

Description

This command refreshes the data for some or all of the resource table records in a result set.

| For the MAS resource table, REFRESH provides data
| only if the MAS was active when the result set was last
| built.

Related commands

COPY, GET, LOCATE, MARK, PERFORM OBJECT, SPECIFY FILTER

Options

ALL
Refreshes all the resource table records in the result set. When you specify ALL:

- The COUNT option is ignored.
- Any records that have been deleted are removed from the result set. Any positions previously held by deleted records are filled in and the remaining records are renumbered. Therefore, the relative position of a given record in a result set may be different after a refresh.

BACKWARD
Refreshes the previous resource table record and continues in a backward direction through the result set refreshing as many records as the COUNT option specifies.

Note: If the record pointer is at the bottom of the result set, using BACKWARD refreshes the current record (which is the last record) and then continues on to previous records.

COUNT*(data-value)*
Specifies the number of resource table records to be refreshed. If you do not specify the COUNT option, only one record is refreshed.

If you do not specify the FORWARD or BACKWARD option, the refresh process moves in a forward direction through the result set.

CURRENT
Refreshes only the current resource table record. When you specify CURRENT, the COUNT option is ignored.

FILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The FILTER option indicates that only those resource table records that meet the specified filter criteria should be considered for refresh.

The number of records that are actually refreshed is determined by the COUNT option. If you do not specify the COUNT option, only the first record that meets the filter criteria is refreshed.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

FORWARD
Refreshes the current resource table record and continues in a forward direction through the result set refreshing as many records as the COUNT option specifies.

MARKED
Indicates that only those resource table records that are marked in the result set should be considered for refresh.

The number of records that are actually refreshed is determined by the COUNT option. If you do not specify the COUNT option, only the first record that is marked is refreshed.

You can mark resource table records by using the MARK and UNMARK commands.

NOTFILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The NOTFILTER option indicates that only those resource table records that do not meet the specified filter criteria should be considered for refresh.

The number of records that are actually refreshed is determined by the COUNT option. If you do not specify the COUNT option, only the first record that does not meet the filter criteria is refreshed.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

NOTMARKED
Indicates that only those resource table records that are not marked in the result set should be considered for refresh.

The number of records that are actually refreshed is determined by the COUNT option. If you do not specify the COUNT option, only the first record that is not marked is refreshed.

You can mark resource table records by using the MARK and UNMARK commands.

NOWAIT
Returns control to your program as soon as the REFRESH command has been accepted, which allows the command to be processed asynchronously.

If you specify the NOWAIT option, you must use a subsequent RECEIVE command to test for the completion of this request. The results of an asynchronous request are returned as ASYNCREQ resource table records. For a complete description of asynchronous processing, see *CICSPlex SM Application Programming Guide*.

Note: If you specify the TOKEN option, the NOWAIT option is assumed by default.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
PERFORM OBJECT.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

TOKEN*(data-value)*
Defines a 1- to 4-character token that you choose to correlate an asynchronous REFRESH request with the result of a subsequent RECEIVE command. This token is for use by your program; CICSPlex SM makes no use of the value. The token is returned by the RECEIVE command when this REFRESH request is complete.

Conditions

The following is a list of the RESPONSE values that can be returned by the REFRESH command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

SCHEDULED
The command has been scheduled for processing.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
| **REQTIMEOUT**	One of the CMASs or MASs to which the request was directed did not respond.
| **SOCRESOURCE**	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

COUNT
FILTER
NOTFILTER
RESULT
THREAD
TOKEN.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CMAS	A CMAS to which the request was directed is not available.

CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.
MAINTPOINT	The maintenance point for the current context is not available.
SCOPE	None of the MASs in the specified scope are available.

NOTPERMIT
A not permitted condition occurred for the following reason:

USRID	The user ID associated with the processing thread does not have the required security authorization.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

TABLEERROR
A resource table record is invalid for the following reason:

DATAERROR	The value associated with one or more resource table attributes is invalid. This error can occur if the resource table is missing required attributes, contains one or more conflicting attributes, or does not exist. Use the FEEDBACK command to retrieve additional data about this error.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

REMOVE

Remove a CICSPlex SM or CICS definition from the data repository.

```
   REMOVE  OBJECT(data-value)  FROM(data-area)  LENGTH(data-value)
                                                                  THREAD(cpsm-token)
    PARM(data-area)  PARMLEN(data-value)     CONTEXT(data-value)
  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This commands removes a CICSPlex SM or CICS definition from the data repository. For definitions that have a CICSplex as their context (such as workload management or real-time analysis definitions), the definition is also removed from the data repositories of all CMASs involved in managing the CICSplex.

Related commands

CREATE, UPDATE

Options

CONTEXT(*data-value*)
Identifies the context for this command. The context must be the 1- to 8-character name of a CMAS or CICSplex.

If you do not specify the CONTEXT option, the default context for the thread is assumed.

FROM(*data-area*)
Identifies a buffer containing a resource table record that represents the definition to be removed. The record must include all of the attributes for the resource table specified on the OBJECT option.

LENGTH(*data-value*)
A fullword value that specifies the length of the FROM buffer.

OBJECT(*data-value*)
Identifies the resource table that represents the definition being removed. This value must be the 1- to 8-character name of a valid CPSM Definition or CICS Definition resource table. For a list of the CICSPlex SM resource tables by type, see *CICSPlex SM Application Programming Guide*.

PARM(*data-area*)
Identifies a buffer containing the parameter expression to be used in removing the definition.

For details on how to use a parameter expression with the REMOVE command, see *CICSPlex SM Application Programming Guide*. For a description of the parameters that are valid for a given resource table, see the *CICSPlex SM Resource Tables Reference*.

PARMLEN(*data-value*)
A fullword value that specifies the length of the PARM buffer.

REASON(*data-ref*)
Names a variable to receive the fullword reason value returned by this command.

RESPONSE(*data-ref*)
Names a variable to receive the fullword response value returned by this command.

THREAD(*cpsm-token*)
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the REMOVE command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
REQTIMEOUT	One of the CMASs to which the request was directed did not respond.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM

An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

CONTEXT
FROM
LENGTH
OBJECT
PARM
PARMLEN
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE

A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CMAS	A CMAS to which the request was directed is not available.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.
MAINTPOINT	The maintenance point for the current context is not available.

NOTPERMIT

A not permitted condition occurred for the following reason:

USRID	The user ID associated with the processing thread does not have the required security authorization.

SERVERGONE

The CMAS to which the processing thread was connected is no longer active.

TABLEERROR

A resource table record is invalid for one of the following reasons:

DATAERROR	The value associated with one or more resource table attributes is invalid. This error can occur if the resource table is missing required attributes, contains one or more conflicting attributes, or does not exist. Use the FEEDBACK command to retrieve additional data about this error.
INVALIDATTR	One of the resource table attributes is invalid.
INVALIDVER	The specified version of the resource table is not supported by CICSPlex SM.

VERSIONINVL

A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

SET

Modify the attributes of one or more resources.

```
     SET  MODIFY(data-area)  LENGTH(data-value)
                                          ALL
                                          CURRENT
                                          FILTER(cpsm-token)
                                          MARKED
                                          NOTFILTER(cpsm-token)
                                          NOTMARKED
                                          POSITION(data-value)

                       FORWARD        NOREFRESH  NOWAIT
      COUNT(data-value)                                         TOKEN(data-value)
                       BACKWARD

    RESULT(cpsm-token)  THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command modifies the attributes of one or more resources as represented by resource table records in an existing result set.

The SET command is valid only for CICS Resource and some CPSM Manager resource tables.

If the context and scope in effect when you issue a SET command include CICS systems that do not support the requested modification, the request is ignored for those CICS systems.

Related commands

COPY, GET, GROUP, LOCATE, MARK, PERFORM OBJECT, PERFORM SET, SPECIFY FILTER

Options

ALL
Modifies all the resource table records in the result set. When you specify ALL, the COUNT option is ignored.

BACKWARD
Modifies the previous resource table record and continues in a backward direction through the result set modifying as many records as the COUNT option specifies.

Note: If the record pointer is at the bottom of the result set, using BACKWARD modifies the current record (which is the last record) and then continues on to previous records.

COUNT*(data-value)*
Specifies the number of resource table records to be modified. If you do not specify the COUNT option, only one record is refreshed.

If you do not specify the FORWARD or BACKWARD option, the modification process moves in a forward direction through the result set.

CURRENT
Modifies only the current resource table record. When you specify CURRENT, the COUNT option is ignored.

FILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The FILTER option indicates that only those resource table records that meet the specified filter criteria should be considered for modification.

The number of records that are actually modified is determined by the COUNT option. If you do not specify the COUNT option, only the first record that meets the filter criteria is modified.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

FORWARD
Modifies the current resource table record and continues in a forward direction through the result set modifying as many records as the COUNT option specifies.

LENGTH*(data-value)*
A fullword value that specifies the length of the MODIFY buffer.

Note: The buffer length you specify should not include any data other than a modification expression.

MARKED
Indicates that only those resource table records that are marked in the result set should be considered for modification.

The number of records that are actually modified is determined by the COUNT option. If you do not specify the COUNT option, only the first record that is marked is modified.

You can mark resource table records by using the MARK and UNMARK commands.

MODIFY*(data-area)*
Identifies a buffer containing the modification expression to be used in modifying the resource table records.

For details on how to form a modification expression, see *CICSPlex SM Application Programming Guide*.

NOREFRESH
Specifies that the resource table records in the source result set should not be refreshed. The records reflect the status of the resources before the SET command was processed.

If you do not specify the NOREFRESH option, the resource table records are refreshed to reflect the resource status after this operation is complete.

NOTFILTER*(cpsm-token)*
Identifies a filter to be used for this operation. The NOTFILTER option indicates that only those resource table records that do not meet the specified filter criteria should be considered for modification.

The number of records that are actually modified is determined by the COUNT option. If you do not specify the COUNT option, only the first record that does not meet the filter criteria is modified.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

NOTMARKED
Indicates that only those resource table records that are not marked in the result set should be considered for modification.

The number of records that are actually modified is determined by the COUNT option. If you do not specify the COUNT option, only the first record that is not marked is modified.

You can mark resource table records by using the MARK and UNMARK commands.

NOWAIT
Returns control to your program as soon as the SET command has been accepted, which allows the command to be processed asynchronously.

If you specify the NOWAIT option, you must use a subsequent RECEIVE command to test for the completion of this request. The results of an asynchronous request are returned as ASYNCREQ resource table records. For a complete description of asynchronous processing, see *CICSPlex SM Application Programming Guide*.

Note: If you specify the TOKEN option, the NOWAIT option is assumed by default.

POSITION*(data-value)*
Modifies the nth resource table record in the result set. When you specify POSITION, the COUNT option is ignored.

This value must be a number that identifies the record's relative position in the result set. The first record in a result set is identified by the number 1.

For example, to modify the fifth resource table record in a result set, you would specify `POSITION(5)`.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
GROUP
PERFORM OBJECT.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

TOKEN*(data-value)*
Defines a 1- to 4-character token that you choose to correlate an asynchronous SET request with the result of a subsequent RECEIVE command. This token is for use by your program; CICSPlex SM makes no use of the value. The token is returned by the RECEIVE command when this SET request is complete.

Conditions

The following is a list of the RESPONSE values that can be returned by the SET command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

SCHEDULED
The command has been scheduled for processing.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.

NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
REQTIMEOUT	One of the CMASs or MASs to which the request was directed did not respond.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDATA
An invalid data error occurred for one of the following reasons:

MODIFY	An attribute value listed in the MODIFY buffer is not valid for the specified attribute.
NOTSUPPORTED	An attribute listed in the MODIFY buffer is not modifiable.

INVALIDPARM
An invalid parameter was detected in either the command string or the MODIFY buffer. The parameter that is invalid is returned as the reason value:

ATTRIBUTE
COUNT
FILTER
LENGTH
MODIFY
NOTFILTER
POSITION
RESULT
THREAD
TOKEN.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CMAS	A CMAS to which the request was directed is not available.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.
MAINTPOINT	The maintenance point for the current context is not available.
SCOPE	Either none of the MASs in the specified scope are available or none of them support the requested modification.

NOTFOUND
A not found condition occurred for one of the following reasons:

ACTION	An action requested in the MODIFY buffer was not found for the specified resource table.
ATTRIBUTE	An attribute specified in the MODIFY buffer was not found for the specified resource table.

NOTPERMIT
A not permitted condition occurred for the following reason:

USRID	The user ID associated with the processing thread does not have the required security authorization.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

TABLEERROR
A resource table record is invalid for one of the following reasons:

DATAERROR	The value associated with one or more resource table attributes is invalid. This error can occur if the resource table is missing required attributes, contains one or more conflicting attributes, or is a duplicate. Use the FEEDBACK command to retrieve additional data about this error.
INVALIDVER	The specified version of the resource table is not supported by CICSPlex SM.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

SPECIFY FILTER

Defines an attribute or value filter and assign an identifying token to it.

```
     SPECIFY  FILTER(data-ref)  CRITERIA(data-area)  LENGTH(data-value)  OBJECT(data-value)
    THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command defines an attribute or value filter and assigns an identifying token to it.

Filters are associated with the specific processing thread on which they are defined; they cannot be shared by other processing threads.

You can define multiple filters for use by a processing thread; CICSPlex SM assigns a unique identifying token to each one.

When a processing thread is terminated, any filters defined by it are discarded.

Related commands

COPY, DELETE, DISCARD, FETCH, GET, GROUP, LISTEN, LOCATE, MARK, PERFORM OBJECT, PERFORM SET, REFRESH, SET, UNMARK

Options

CRITERIA(*data-area*)
Identifies a buffer containing the filter expression to be used for this operation. The CRITERIA option filters only those resource table records that meet the specified criteria.

For details on how to form a filter expression, see *CICSPlex SM Application Programming Guide*.

Note: You cannot specify the EYU_CICSNAME or EYU_CICSREL attributes in a filter expression.

FILTER(*data-ref*)
Names a variable to receive the token that CICSPlex SM assigns to this filter.

This identifying token must be specified on all subsequent commands that use this filter.

LENGTH(*data-value*)
A fullword value that specifies the length of the CRITERIA buffer.

Note: The buffer length you specify should not include any data other than a filter expression.

OBJECT(*data-value*)
Identifies the resource table for which a filter is being created. This value must be the 8-character name of a valid resource table.

REASON(*data-ref*)
Names a variable to receive the fullword reason value returned by this command.

RESPONSE(*data-ref*)
Names a variable to receive the fullword response value returned by this command.

THREAD(*cpsm-token*)
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the SPECIFY FILTER command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDATA
Invalid data was detected. The parameter that contains invalid data is returned as the reason value:

CRITERIA	An attribute value listed in the CRITERIA buffer is not valid for the specified attribute.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

CRITERIA

FILTER
LENGTH
OBJECT
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

NOTFOUND
A not found condition occurred for the following reason:

ATTRIBUTE	An attribute specified in the CRITERIA buffer was not found for the specified resource table.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

SPECIFY VIEW

Build a customized view of a given resource table.

```
    SPECIFY  VIEW(data-value)  FIELDS(data-area)  LENGTH(data-value)  OBJECT(data-value)
   THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command builds a customized view of a given resource table.

Views can be built only for resource tables with a type of CICS Resource.

Views are associated with the specific processing thread on which they are built; they cannot be shared by other processing threads.

When a processing thread is terminated, any views built by it are deleted.

| The name you assign to a view takes precedence over any existing resource table names. You can redefine an existing resource table name to represent a customized view of that resource table.

| You are recommended to use names for customized views that are not already assigned either to other customized views or to CICSPlex SM-supplied resource tables. If you do use a name that is already assigned, you should be aware that your processing could be affected. For more details, see *CICSPlex SM Application Programming Guide*.

| If and when you migrate to a later version of CICSPlex SM, you should check that any new resource tables do not have the same names as any customized views. For more details, see *CICSPlex SM Application Programming Guide*.

Related commands

DISCARD, GET

Options

FIELDS*(data-area)*
Identifies a buffer containing the order expression to be used for this operation.

For details on how to use an order expression with the SPECIFY VIEW command, see *CICSPlex SM Application Programming Guide*.

Note: You cannot specify the EYU_CICSNAME or EYU_CICSREL attributes in an order expression.

LENGTH*(data-value)*
A fullword value that specifies the length of the FIELDS buffer.

Note: The buffer length you specify should not include any data other than an order expression.

OBJECT*(data-value)*
Identifies the resource table for which a view is being created. This value must be the 1- to 8-character name of a valid CICS Resource table. For a list of the CICSPlex SM resource tables by type, see *CICSPlex SM Application Programming Guide*.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

VIEW*(data-value)*
| Defines a 1- to 8-character name for the view being built.

Conditions

The following is a list of the RESPONSE values that can be returned by the SPECIFY VIEW command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

DUPE
A duplicate condition occurred for the following reason:

VIEW	The specified view already exists and cannot be built.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

| **SOCRESOURCE**	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected in either the command string or the FIELDS buffer. The parameter that is invalid is returned as the reason value:

ATTRIBUTE
FIELDS
LENGTH
OBJECT
THREAD
VIEW.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

NOTFOUND
A not found condition occurred for the following reason:

ATTRIBUTE	An attribute specified in the FIELDS buffer was not found for the specified resource table.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

TABLEERROR
A resource table record is invalid for one of the following reasons:

DATAERROR	The value associated with one or more resource table attributes is invalid. This error can occur if the resource table is missing required attributes, contains one or more conflicting attributes, or does not exist. Use the FEEDBACK command to retrieve additional data about this error.
INVALIDVER	The specified version of the resource table is not supported by CICSPlex SM.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

TERMINATE

Terminate all API processing on all active threads.

```
      TERMINATE  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command terminates all API processing on all active threads created by the CICS or MVS/ESA task that issues the command.

Issuing TERMINATE is equivalent to issuing the DISCONNECT command for each active thread individually.

Any resources that are associated with the thread are released, including result sets, filters, views, diagnostic data, and outstanding asynchronous requests.

Related commands

CONNECT, DISCONNECT

Options

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

Conditions

The following is a list of the RESPONSE values that can be returned by the TERMINATE command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

TRANSLATE

Translate resource table attribute values.

```
      TRANSLATE  OBJECT(data-value)  ATTRIBUTE(data-value)  FROMCV(data-value)  TOCHAR(data-ref)
                                                            FROMCHAR(data-value)  TOCV(data-ref)

     THREAD(cpsm-token)  RESPONSE(data-ref)  REASON(data-ref)
```

Description

This command translates resource table attribute values that are maintained in an encoded form (such as EYUDA and CVDA values) between their internal coded format and an external display format.

If your program is written in REXX, you can use the TPARSE command to access a resource table record and translate its attribute values. However, if you use the ASIS option with TPARSE, attribute values are not translated into their external format; in that case, you would need to use TRANSLATE after using TPARSE to receive the formatted display values. For a description of the TPARSE command, see Chapter 3, “REXX functions and commands” on page 87.

In some CICS environments, the DFHVALUE function returns incompatible CVDA values for the following resource table attributes:

Resource table	Attribute value	CICS Environment
CONNECT	RECOVSTATUS(NRS)	CICS/VSE
LOCTRAN	RESSEC(RESSECEXT)	CICS/VSE CICS/MVS
PROGRAM	LPASTAT(SVA)	CICS/VSE
PROGRAM	LPASTAT(NOTSVA)	CICS/VSE

Because these CVDA values conflict with values used in other CICS environments, CICSPlex SM must modify them to retain their uniqueness. CICSPlex SM adds 9000 to the value returned by DFHVALUE for each of these CICS CVDA attributes.

If you want to translate any of these attributes in a CICS environment, you must add 9000 to the value you received from DFHVALUE before presenting the attribute to CICSPlex SM.

Options

ATTRIBUTE(*data-value*)
Identifies the resource table attribute that is to be translated. This value must be the 1- to 12-character name of a valid attribute for the resource table.

FROMCHAR(*data-value*)
Specifies the 1- to 12-character value for the specified attribute.

FROMCV(*data-value*)
Specifies the 4-byte internal coded value for the specified attribute.

OBJECT(*data-value*)
Identifies the resource table to which the attribute being translated belongs. This value must be the 8-character name of a valid resource table.

REASON(*data-ref*)
Names a variable to receive the fullword reason value returned by this command.

RESPONSE(*data-ref*)
Names a variable to receive the fullword response value returned by this command.

THREAD(*cpsm-token*)
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

TOCHAR(*data-ref*)
Names a variable to receive the result of translating an internal coded value to the 1- to 12-character value for the specified attribute.

TOCV(*data-ref*)
Names a variable to receive the result of translating a character value to the 4-byte internal coded value for the specified attribute.

Conditions

The following is a list of the RESPONSE values that can be returned by the TRANSLATE command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

ATTRIBUTE
FROMCHAR
FROMCV
OBJECT
THREAD
TOCHAR
TOCV.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

TABLEERROR
A resource table record is invalid for the following reason:

INVALIDVER	The specified version of the resource table is not supported by CICSPlex SM.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

UNMARK

Remove the marks placed on resource table records.

```
          CURRENT
 UNMARK
          ALL                                        COUNT(data-ref)
           FILTER(cpsm-token)
          NOTFILTER(cpsm-token)
          POSITION(data-value)
          PARM(data-area)  PARMLEN(data-value)

                                           RESULT(cpsm-token)  THREAD(cpsm-token)  RESPONSE(data-ref
  INTO(data-area)  LENGTH(data-ref)

 REASON(data-ref)
```

Description

This command removes the marks placed on resource table records by a previous MARK command. The UNMARK command always begins processing with the last record that was fetched, rather than the next one in the result set.

Related commands

LOCATE, MARK

Options

ALL
Removes the marks from all resource table records in the result set.

COUNT(*data-ref*)
Names a variable to receive the number of resource table records that could not be unmarked.

CURRENT
Removes the mark from only the current resource table record.

FILTER(*cpsm-token*)
Identifies a filter to be used for this operation. The FILTER option removes the marks from only those resource table records that meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

INTO(*data-area*)
Identifies a buffer to receive a list of resource table records that could not be unmarked.

This buffer must be long enough to hold the maximum number of record numbers that could result from your UNMARK request (in the event that none of them can be unmarked). Record numbers are listed individually (not by range) in the INTO buffer and are separated by commas.

Note: If you receive a RESPONSE value of WARNING AREATOOSMALL (because the buffer was not long enough), the data returned in this buffer represents a partial list of the records that could not be unmarked.

LENGTH(*data-ref*)
A fullword value that specifies the length of the INTO buffer.

The value that CICSPlex SM returns in this field depends on the RESPONSE value for the UNMARK command:

OK The actual length of the data returned in the INTO buffer.

WARNING AREATOOSMALL
The buffer length that would be required to hold a complete list of records that could not be unmarked.

NOTFILTER(*cpsm-token*)
Identifies a filter to be used for this operation. The NOTFILTER option removes the marks from only those resource table records that do not meet the specified filter criteria.

The *cpsm-token* value that identifies a filter is returned by the SPECIFY FILTER command.

PARM(*data-area*)
Identifies a buffer containing the parameter expression that lists the resource table records to be unmarked.

The parameter expression for the UNMARK command is a character string of record numbers. For example:

```
PARM('1,3,6:9,24.')
```

To specify individual records, separate the record numbers with a comma. To specify a range of records, separate the low and high record numbers with a colon. The whole parameter expression must end with a period.

For details on how to use a parameter expression with the UNMARK command, see *CICSPlex SM Application Programming Guide*.

PARMLEN*(data-value)*
A fullword value that specifies the length of the PARM buffer.

POSITION*(data-value)*
Removes the mark from the nth resource table record in the result set.

This value must be a number that identifies the record's relative position in the result set. The first record in a result set is identified by the number 1.

For example, to unmark the fifth resource table record in a result set, you would specify `POSITION(5)`.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

COPY
GET
GETDEF
GROUP
PERFORM OBJECT.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the UNMARK command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

NODATA
No records were found that matched the specified search criteria.

WARNING
The command completed processing with a warning, for one of the following reasons:

AREATOOSMALL	You specified the INTO and LENGTH options, but the buffer was not long enough to hold the string of records that could not be unmarked.
DATAERROR	One or more of the records specified in the PARM buffer could not be found to be unmarked. If you specified the COUNT option, the number of records that could not be unmarked is returned. If you specified the INTO and LENGTH options, a list of the records is returned in the buffer.

BUSY
A busy condition occurred for the following reason:

RESULT	The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.
SOLRESOURCE	A required resource that is locally owned (that is, owned by the address space where the processing thread is running) is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

COUNT
FILTER
INTO
LENGTH
NOTFILTER
PARM
PARMLEN
RESULT
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.
NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

UPDATE

Update an existing CICSPlex SM or CICS definition.

```
     UPDATE  OBJECT(data-value)    FROM(data-area)
                                             PARM(data-area)  PARMLEN(data-value)
                               RESULT(cpsm-token)  MODIFY(data-area)
    LENGTH(data-value)                           THREAD(cpsm-token)  RESPONSE(data-ref)
                        CONTEXT(data-value)
    REASON(data-ref)
```

Description

This command updates an existing CICSPlex SM or CICS definition according to the attribute values you specify.

The updated definition replaces the existing definition in the CICSPlex SM data repository.

For definitions that have a CICSplex as their context (such as workload management or real-time analysis definitions), the definition is also updated in the data repositories of all CMASs involved in managing the CICSplex.

Related commands

CREATE, REMOVE

Options

CONTEXT*(data-value)*
Identifies the context for this command. The context must be the 1- to 8-character name of a CMAS or CICSplex.

If you do not specify the CONTEXT option, the default context for the thread is assumed.

FROM*(data-area)*
Identifies a buffer containing a resource table record that represents the definition to be updated.

The record must include all of the updates for all of the attributes of the resource table specified on the OBJECT option. For optional attributes that you do not want to specify, set the field to the appropriate null value for the attribute's data type.

LENGTH*(data-value)*
A fullword value that specifies the length of the FROM or MODIFY buffer.

Note: The buffer length you specify should not include any data other than a resource table record or modification expression.

MODIFY*(data-area)*
Identifies a buffer containing the modification expression to be used in modifying CICS Definition resource table records.

For details on how to form a modification expression, see *CICSPlex SM Application Programming Guide*.

OBJECT*(data-value)*
Identifies the resource table that represents the definition being updated. This value must be the 8-character name of a valid CPSM Definition or CICS Definition resource table. For a list of the CICSPlex SM resource tables by type, see *CICSPlex SM Application Programming Guide*.

PARM*(data-area)*
Identifies a buffer containing a parameter expression to be used in updating the definition.

For details on how to use a parameter expression with the UPDATE command, see *CICSPlex SM Application Programming Guide*. For a description of the parameters that are valid for a given resource table, see the *CICSPlex SM Resource Tables Reference*.

PARMLEN*(data-value)*
A fullword value that specifies the length of the PARM buffer.

REASON*(data-ref)*
Names a variable to receive the fullword reason value returned by this command.

RESPONSE*(data-ref)*
Names a variable to receive the fullword response value returned by this command.

RESULT*(cpsm-token)*
Identifies the API result set to be processed by this operation. The result set must contain CICS Definition resource table records. The records are updated according to the modification expression you supply in the MODIFY buffer.

The result set can be one produced by any of these commands:

COPY
GET
GROUP
PERFORM OBJECT.

THREAD_(cpsm-token)_
Identifies the API thread to be used for this operation. The _cpsm-token_ value that identifies a thread is returned by the CONNECT command.

Conditions

The following is a list of the RESPONSE values that can be returned by the UPDATE command. The description of each RESPONSE includes a list of associated REASON values, if appropriate.

OK
The command completed processing successfully.

ENVIRONERROR
An environment error occurred for one of the following reasons:

NOSERVICE	The application stub program could not load the API service module.
NOSTORAGE	The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
REQTIMEOUT	One of the CMASs or MASs to which the request was directed did not respond.
SOCRESOURCE	A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:

ABENDED	Command processing abended.
EXCEPTION	Command processing encountered an exceptional condition.

INVALIDATA
An invalid data error occurred for one of the following reasons:

MODIFY	An attribute value listed in the MODIFY buffer is not valid for the specified attribute.
NOTSUPPORTED	An attribute listed in the MODIFY buffer is not modifiable.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:

CONTEXT
FROM
LENGTH
MODIFY
OBJECT
PARM
PARMLEN
RESULT
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:

APITASK	The API control subtask is not active.
CMAS	A CMAS to which the request was directed is not available.
CPSMAPI	The CMAS to which the processing thread is connected is not available for API processing.
MAINTPOINT	The maintenance point for the current context is not available.

NOTPERMIT
A not permitted condition occurred for the following reason:

USRID	The user ID associated with the processing thread does not have the required security authorization.

SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

TABLEERROR
A resource table record is invalid for one of the following reasons:

DATAERROR	The value associated with one or more resource table attributes is invalid. This error can occur if: The resource table is missing required attributes, contains one or more conflicting attributes, or does not exist. A CICS resource definition contains attributes that would cause the EXEC CICS CREATE command to issue warnings. Use the FEEDBACK command to retrieve additional data about this error.
INVALIDATTR	One of the resource table attributes is invalid.
INVALIDVER	The specified version of the resource table is not supported by CICSPlex SM.

VERSIONINVL
A version conflict occurred for one of the following reasons:

NOTSUPPORTED	The version of the application stub program used for this command is not supported.

NOTVSNCONN	The version of the application stub program used for this command is not the same as the version used with the CONNECT command.

Chapter 3. REXX functions and commands

This chapter contains detailed descriptions of the REXX functions and commands supplied with CICSPlex SM. These functions and commands can be used only with the REXX run-time interface.

| The REXX functions supplied with CICSPlex SM make use of standard REXX variable substitution rules. In addition to REXX return codes, these functions can produce EYUARnnnn messages. For descriptions of those messages, see *CICSPlex SM Messages and Codes*.

| Each description includes the following

- A description of the command
- Purpose
- Syntax of command (*var* represents a variable)
- Available options for the command
- REXX response codes returned by the command

| The functions are presented in alphabetical order:

- "EYUAPI()"
- "EYUINIT()" on page 88
- "EYUREAS()" on page 88
- "EYURESP()" on page 88
- "EYUTERM()" on page 89

Functions

EYUAPI()

Passes an API command to CICSPlex SM.

```
   var = EYUAPI(command string)

      OR

   var = EYUAPI('command string')
```

Description: This function passes an API command to CICSPlex SM. You must issue an EYUAPI or EYUINIT function before you can use the ADDRESS CPSM command to pass API commands to REXX.

Options:

command string
Identifies the API command and options to be passed.

Return codes: The following is a list of the REXX return codes that can be returned by the EYUAPI function in its assigned variable (*var*).

These return codes indicate what REXX did with the EYUAPI function; they do not indicate whether the API command that was passed was successfully processed by CICSPlex SM. For that information, you must refer to the RESPONSE and REASON values returned by the command.

0	The EYUAPI function was successful.
1	The EYUAPI function failed.

EYUINIT()

Initialize the CICSPlex SM API environment and allocate the necessary resources.

```
   var = EYUINIT()
```

Description: This command initializes the CICSPlex SM API environment and allocates the necessary resources. EYUINIT should be the first function issued in a REXX program.

Note: You must issue an EYUINIT or EYUAPI function before you can use the ADDRESS CPSM command to pass API commands to REXX.

Return codes: The following is a list of the REXX return codes that can be returned by the EYUINIT function in its assigned variable (*var*).

0	The EYUINIT function was successful.
1	The EYUINIT function failed.

EYUREAS()

Translate the numeric value returned by the REASON option of an API command.

```
   var = EYUREAS(reason)
```

Description: This command translates the numeric value returned by the REASON option of an API command into its character equivalent and vice versa.

Options:

reason
Is the REASON value to be translated.

Return codes: The following is a list of the REXX return codes that can be returned by the EYUREAS function in its assigned variable (*var*).

nnnn	The numeric or character equivalent of the specified REASON value.
−1	The specified REASON value is invalid and could not be translated.

EYURESP()

Translate the numeric value returned by the RESPONSE option of an API command.

```
   var = EYURESP(response)
```

Description: This command translates the numeric value returned by the RESPONSE option of an API command into its character equivalent and vice versa.

Options

response
Is the RESPONSE value to be translated.

Return codes: The following is a list of the REXX return codes that can be returned by the EYURESP function in its assigned variable (*var*).

nnnn	The numeric or character equivalent of the specified RESPONSE value.
−1	The specified RESPONSE value is invalid and could not be translated.

EYUTERM()

Terminate the CICSPlex SM API environment and release any allocated resources.

```
var = EYUTERM()
```

Description: This command terminates the CICSPlex SM API environment and releases any allocated resources. EYUTERM should be the last function issued in a REXX program.

Note: If the CICSPlex SM host subcommand environment is actually installed at your enterprise (as opposed to being called from the function package), you may not need to use EYUTERM at the end of every program. The resources that remain allocated can be reused by the next program that accesses the host subcommand environment.

Return codes: The following is a list of the REXX return codes that can be returned by the EYUTERM function in its assigned variable (*var*).

0 The EYUTERM function was successful.

1 The EYUTERM function failed.

Commands

The REXX-specific commands supplied with CICSPlex SM perform a series of API commands internally and return the results to REXX.

The commands are presented here in alphabetical order. Each description includes the purpose, syntax, and available options for the command.

Notes:

1. You cannot use these commands to process user-defined views of a resource table that were created by the SPECIFY VIEW command. If you create a view with the same name as a supplied resource table and then specify that name on one of these commands, the command fails.
2. These commands do not use the RESPONSE and REASON options. The result of these REXX-specific processes is returned by the STATUS option.
3. These commands do not provide any useful FEEDBACK information. The API commands that are issued internally reuse the same feedback area. So, when one of these commands finishes processing, the feedback area does not represent the entire sequence of events.

The commands are:

"TBUILD" on page 91
"TPARSE" on page 92

TBUILD

Build a resource table record from a set of variables.

```
      TBUILD  OBJECT(data-value)  PREFIX(data-value)  STATUS(data-ref)  VAR(data-area)
                                                                            ASIS

     THREAD(cpsm-token)
```

Description: This command builds a resource table record from a set of variables that represent the individual attributes of a CICSPlex SM or CICS definition. A definition is represented by a resource table with a type of CPSM Definition or CICS Definition.

You form the attribute variables by adding a prefix to the attribute name, like this:

```
  prefix_attribute
```

where `prefix` is a text string that you supply and `attribute` is the name of an attribute in the resource table. You must insert an underscore character (_) between the prefix and the attribute name.

The resource table record can be placed in any valid REXX variable, including a stem variable.

TBUILD only uses the attributes that you specify; it does not assume any default values for optional attributes. If you do not supply a variable for an attribute that is optional, the corresponding field in the resource table record is initialized according to its data type (that is, character fields are set to blanks, binary data and EYUDA values are set to zeroes).

Note: For a list of the CICSPlex SM resource tables by type, see *CICSPlex SM Application Programming Guide*. For a complete description of a particular resource table and its attributes, see the *CICSPlex SM Resource Tables Reference*.

Options:

ASIS
Indicates that the resource table attribute values are already in their internal format; they are to be processed as is, rather than translated.

You must use the ASIS option to rebuild a CICSPlex SM or CICS definition that you previously parsed (with the TPARSE ASIS command).

OBJECT*(data-value)*
Identifies the resource table for which a record is to be built. This value must be the 1- to 8-character name of a valid CPSM Definition or CICS Definition resource table.

Note: You cannot use the TBUILD command to process a resource table view that was created by the SPECIFY VIEW command. If you create a view with the same name as a supplied resource table and then specify that name on a TBUILD command, the command fails.

PREFIX*(data-value)*
Specifies the prefix you used to name the variables that contain the resource table attributes.

Note: The maximum allowable length for a prefix is determined by REXX and the environment in which the program runs.

STATUS*(data-ref)*
Names a variable to receive the REXX status value returned for this command. The status is returned in character form as one of the following:

OK
The TBUILD command completed processing successfully.

SYNTAX ERROR
The TBUILD command could not be processed because of a syntax error. EYUARnnnn messages that describe the error are written to the destination defined on your system for IRXSAY WRITEERR output.

FAILURE
The TBUILD command failed because some of the data it was attempting to process is invalid. Trace data is written to a REXX stem variable called EYUTRACE. EYUARnnnn messages that describe the failure may also be written to the destination defined on your system for IRXSAY WRITEERR output.

Note: For more information about the EYUTRACE stem variable, see *CICSPlex SM Application Programming Guide*.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

VAR*(data-area)*
Names a variable to receive the resource table record that is built by TBUILD.

TPARSE

Parse a resource table record from a variable into a set of variables.

```
      TPARSE  OBJECT(data-value)  PREFIX(data-value)  STATUS(data-ref)  VAR(data-area)
                                                                            ASIS

     THREAD(cpsm-token)
```

Description: This command parses a resource table record from a variable into a set of variables that represent the individual attributes of the table. You can use TPARSE with any type of CICSPlex SM resource table.

The resource table variable can be any valid REXX variable, including a stem variable. The output variables are formed by adding a prefix to the attribute name, like this:

```
prefix_attribute
```

where `prefix` is a text string that you supply and `attribute` is the name of an attribute in the resource table. An underscore (_) is inserted between the prefix and the attribute name.

Note: For complete descriptions of the resource tables and their attributes, see the *CICSPlex SM Resource Tables Reference*.

Options:

ASIS
Specifies that the resource table attribute values are not to be translated into their external format; they are to be returned as is. Attribute values are presented as follows:

Character values have trailing blanks.
Binary values have leading zeroes and are not converted to display format.
EYUDA and CVDA values are not converted to character format.

You must use the ASIS option to parse a CPSM Definition or CICS Definition resource table that you want to rebuild (with the TBUILD ASIS command).

Note: If you use the ASIS option with EYUDA or CVDA values, you can use the TRANSLATE command to convert the coded numeric value into a character value.

OBJECT*(data-value)*
Identifies the resource table that is to be parsed. This value must be the 1- to 8-character name of a valid resource table.

Note: You cannot use the TPARSE command to process a resource table view that was created by the SPECIFY VIEW command. If you create a view with the same name as a supplied resource table and then specify that name on a PARSE command, the command fails.

PREFIX*(data-value)*
Specifies the prefix you want to use to name the attribute variables returned by TPARSE.

Note: The maximum allowable length for a prefix is determined by REXX and the environment in which the program runs.

STATUS*(data-ref)*
Names a variable to receive the REXX status value returned for this command. The status is returned in character form as one of the following:

OK
The TPARSE command completed processing successfully.

SYNTAX ERROR
The TPARSE command could not be processed because of a syntax error. EYUARnnnn messages that describe the error are written to the destination defined on your system for IRXSAY WRITEERR output.

FAILURE
The TPARSE command failed because some of the data it was attempting to process is invalid. Trace data is written to a REXX stem variable called EYUTRACE. EYUARnnnn messages that describe the failure may also be written to the destination defined on your system for IRXSAY WRITEERR output.

Note: For more information about the EYUTRACE stem variable, see *CICSPlex SM Application Programming Guide*.

THREAD*(cpsm-token)*
Identifies the API thread to be used for this operation. The *cpsm-token* value that identifies a thread is returned by the CONNECT command.

VAR*(data-area)*
Names a variable that contains the resource table record to be parsed.

Appendix A. RESPONSE and REASON values

This appendix provides a summary of the RESPONSE and REASON values returned by each API command.

For descriptions of these values, refer to the description of the command that returns them. For a list of RESPONSE and REASON character values and their numeric equivalents, see Appendix B, “EYUDA values” on page 99. For a discussion of the RESPONSE and REASON options, see *CICSPlex SM Application Programming Guide*.

COMMAND	RESPONSE	REASONS
ADDRESS		
	OK	
	ENVIRONERROR	NOSERVICE, NOSTORAGE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	ECB, SENTINEL, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
CANCEL		
	OK	
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	NOTIFICATION, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
CONNECT		
	OK	
	ENVIRONERROR	APITASKERR, NOSERVICE, NOSTORAGE, SOCRESOURCE, SOERESOURCE, SOLRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	CONTEXT, SCOPE, SIGNONPARM, USRID, VERSION
	NOTAVAILABLE	APITASK, CPSMAPI, CPSMSERVER, CPSMSYSTEM, CPSMVERSION
	NOTPERMIT	USRID
	VERSIONINVL	NOTSUPPORTED
COPY		
	OK	
	NODATA	
	BUSY	FROM, TO
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INCOMPATIBLE	INVALIDOBJ
	INVALIDPARM	FILTER, FROM, NOTFILTER, THREAD, TO
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
CREATE		
	OK	
	ENVIRONERROR	NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	CONTEXT, FROM, LENGTH, OBJECT, PARM, PARMLEN, THREAD
	NOTAVAILABLE	APITASK, CMAS, CPSMAPI, MAINTPOINT
	NOTPERMIT	USRID
	SERVERGONE	
	TABLEERROR	DATAERROR, INVALIDATTR, INVALIDVER
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
DELETE		
	OK	
	NODATA	
	BUSY	RESULT

COMMAND	RESPONSE	REASONS
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE, SOLRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	FILTER, NOTFILTER, RESULT, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
DISCARD		
	OK	
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INUSE	FILTER, VIEW
	INVALIDPARM	FILTER, RESULT, THREAD, VIEW
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
DISCONNECT		
	OK	
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
FEEDBACK		
	OK	
	NODATA	
	WARNING	AREATOOSMALL
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	COUNT, INTO, LENGTH, RESULT, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
FETCH		
	OK	
	NODATA	BACKWARD, FORWARD
	WARNING	AREATOOSMALL
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE, SOLRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	COUNT, FILTER, INTO, LENGTH, NOTFILTER, POSITION, RESULT, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
GET		
	OK	
	SCHEDULED	
	NODATA	
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDATA	CRITERIA
	INVALIDPARM	CONTEXT, CRITERIA, FILTER, LENGTH, OBJECT, PARM, PARMLEN, RESULT, SCOPE, THREAD, TOKEN
	NOTAVAILABLE	APITASK, CMAS, CPSMAPI, MAINTPOINT, SCOPE
	NOTFOUND	ATTRIBUTE
	NOTPERMIT	USRID
	SERVERGONE	
	TABLEERROR	DATAERROR
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN

COMMAND	RESPONSE	REASONS
GETDEF		
	OK	
	NODATA	
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INCOMPATIBLE	INVALIDOBJ
	INVALIDPARM	ATTRIBUTE, OBJECT, RESOURCE, RESULT, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	TABLEERROR	DATAERROR, INVALIDVER
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
GROUP		
	OK	
	NODATA	
	BUSY	FROM, TO
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	BY, FILTER, FROM, LENGTH, NOTFILTER, SUMOPT, THREAD, TO
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
LISTEN		
	OK	
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INCOMPATIBLE	INVALIDEVT
	INVALIDPARM	CONTEXT, EVENT, FILTER, NOTFILTER, NOTIFICATION, THREAD, TOKEN
	NOTAVAILABLE	APITASK, CPSMAPI, PLEXMGR
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
LOCATE		
	OK	
	NODATA	BACKWARD, FORWARD
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE, SOLRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	BACKWARD, FILTER, FORWARD, NOTFILTER, POSITION, RESULT, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
MARK		
	OK	
	NODATA	
	WARNING	AREATOOSMALL, DATAERROR
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE, SOLRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	COUNT, FILTER, INTO, LENGTH, NOTFILTER, PARM, PARMLEN, RESULT, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
ORDER		
	OK	
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	BY, LENGTH, RESULT, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	

COMMAND	RESPONSE	REASONS
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
PERFORM OBJECT		
	OK	
	SCHEDULED	
	NODATA	BACKWARD, FORWARD
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDATA	PARM, CRITERIA
	INVALIDPARM	ACTION, CONTEXT, CRITERIA, FILTER, LENGTH, OBJECT, PARM, PARMLEN, RESULT, SCOPE, THREAD, TOKEN
	NOTAVAILABLE	APITASK, CMAS, CPSMAPI, MAINTPOINT, PLEXMGR, SCOPE
	NOTFOUND	ACTION, ATTRIBUTE
	NOTPERMIT	USRID
	SERVERGONE	
	TABLEERROR	DATAERROR
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
PERFORM SET		
	OK	
	SCHEDULED	
	NODATA	BACKWARD, FORWARD
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, REQTIMEOUT SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDATA	PARM, CRITERIA
	INVALIDPARM	ACTION, FILTER, NOTFILTER, PARM, PARMLEN, POSITION, RESULT, THREAD, TOKEN
	NOTAVAILABLE	APITASK, CMAS, CPSMAPI, MAINTPOINT, PLEXMGR, SCOPE
	NOTFOUND	ACTION, ATTRIBUTE
	NOTPERMIT	USRID
	SERVERGONE	
	TABLEERROR	DATAERROR
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
QUALIFY		
	OK	
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	CONTEXT, SCOPE, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI, PLEXMGR
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
QUERY		
	OK	
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	CONTEXT, DATALENGTH, OBJECT, RESULT, THREAD, TYPE
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
RECEIVE		
	OK	
	NODATA	
	WARNING	AREATOOSMALL
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	DELAY, INTO, LENGTH, OBJECT, THREAD, TOKEN
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
REFRESH		
	OK	

COMMAND	RESPONSE	REASONS
	SCHEDULED	
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	COUNT, FILTER, NOTFILTER, RESULT, THREAD, TOKEN
	NOTAVAILABLE	APITASK, CMAS, CPSMAPI, MAINTPOINT, SCOPE
	NOTPERMIT	USRID
	SERVERGONE	
	TABLEERROR	DATAERROR
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
REMOVE		
	OK	
	ENVIRONERROR	NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	CONTEXT, FROM, LENGTH, OBJECT, PARM, PARMLEN, THREAD
	NOTAVAILABLE	APITASK, CMAS, CPSMAPI, MAINTPOINT
	NOTPERMIT	USRID
	SERVERGONE	
	TABLEERROR	DATAERROR, INVALIDATTR, INVALIDVER
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
SET		
	OK	
	SCHEDULED	
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDATA	MODIFY, NOTSUPPORTED
	INVALIDPARM	ATTRIBUTE, COUNT, FILTER, LENGTH, MODIFY, NOTFILTER, POSITION, RESULT, THREAD, TOKEN
	NOTAVAILABLE	APITASK, CMAS, CPSMAPI, MAINTPOINT, SCOPE
	NOTFOUND	ACTION, ATTRIBUTE
	NOTPERMIT	USRID
	SERVERGONE	
	TABLEERROR	DATAERROR, INVALIDVER
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
SPECIFY FILTER		
	OK	
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	CRITERIA, FILTER, LENGTH, OBJECT, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	NOTFOUND	ATTRIBUTE
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
SPECIFY VIEW		
	OK	
	DUPE	VIEW
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDATA	CRITERIA
	INVALIDPARM	ATTRIBUTE, FIELDS, LENGTH, OBJECT, THREAD, VIEW
	NOTAVAILABLE	APITASK, CPSMAPI
	NOTFOUND	ATTRIBUTE
	SERVERGONE	
	TABLEERROR	DATAERROR, INVALIDVER
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
TERMINATE		
	OK	
	ENVIRONERROR	NOSERVICE, NOSTORAGE
	FAILED	ABENDED, EXCEPTION
TRANSLATE		
	OK	

COMMAND	RESPONSE	REASONS
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	ATTRIBUTE, FROMCHAR, FROMCV, OBJECT, THREAD, TOCHAR, TOCV
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	TABLEERROR	INVALIDVER, INVALIDVER
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
UNMARK		
	OK	
	NODATA	
	WARNING	AREATOOSMALL, DATAERROR
	BUSY	RESULT
	ENVIRONERROR	NOSERVICE, NOSTORAGE, SOCRESOURCE, SOLRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	COUNT, FILTER, INTO, LENGTH, NOTFILTER, PARM, PARMLEN, RESULT, THREAD
	NOTAVAILABLE	APITASK, CPSMAPI
	SERVERGONE	
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN
UPDATE		
	OK	
	ENVIRONERROR	NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE
	FAILED	ABENDED, EXCEPTION
	INVALIDPARM	CONTEXT, FROM, LENGTH, MODIFY, OBJECT, PARM, PARMLEN, RESULT, THREAD
	NOTAVAILABLE	APITASK, CMAS, CPSMAPI, MAINTPOINT
	NOTPERMIT	USRID
	SERVERGONE	
	TABLEERROR	DATAERROR, INVALIDATTR, INVALIDVER
	VERSIONINVL	NOTSUPPORTED, NOTVSNCONN

Appendix B. EYUDA values

This appendix lists the CICSPlex SM API EYUDA values and their numeric equivalents:

RESPONSE	Values returned by the RESPONSE option of an API command. These EYUDAs have numeric values in the range of 1024 – 1279. See "EYUDA RESPONSE values in numerical order" on page 100.
REASON	Values returned by the REASON option of an API command. These EYUDAs have numeric values in the range of 1280 – 1370. See "EYUDA REASON values in numerical order" on page 100.

EYUDA RESPONSE values in numerical order

This section lists the RESPONSE EYUDAs in numerical order.

EYUDA	Value
OK	1024
SCHEDULED	1025
NOTFOUND	1026
NODATA	1027
INVALIDPARM	1028
FAILED	1029
ENVIRONERROR	1030
NOTPERMIT	1031
BUSY	1032
SERVERGONE	1033
NOTAVAILABLE	1034
VERSIONINVL	1035
INVALIDCMD	1036
WARNING	1037
TABLEERROR	1038
INCOMPATIBLE	1039
INUSE	1040
INVALIDATA	1041
DUPE	1042

EYUDA REASON values in numerical order

This section lists the REASON EYUDAs in numerical order.

EYUDA	Value
THREAD	1280
OBJECT	1281
CONTEXT	1282
RESULT	1283
COUNT	1284
LENGTH	1285
FILTER	1286
NOTFILTER	1287
FORWARD	1288
BACKWARD	1289
POSITION	1290
DELAY	1291
NOTIFICATION	1292
SIGNONPARM	1293
SCOPE	1294
RESOURCE	1295
FROM	1296
TO	1297
INTO	1298
CRITERIA	1299
BY	1300
ACTION	1301
ECB	1302
SENTINEL	1303
FEEDBACK	1304
VENT	1305
TOKEN	1306
MODIFY	1307
VIEW	1308
FIELDS	1309
ATTRIBUTE	1310
FROMCV	1311
TOCHAR	1312
FROMCHAR	1313
TOCV	1314
PARM	1315
PARMLEN	1316
SUMOPT	1317
TYPE	1318
DATALENGTH	1319
SOLRESOURCE	1320
SOCRESOURCE	1321
SOERESOURCE	1322
MAINTPOINT	1323
SYSNOTACT	1324
SYSLVLBAD	1325
SYSNOTLCL	1326
CICSRELBAD	1327
ARMNOTREG	1328
ARMNOTACT	1329
ARMPOLCHK	1330
ABENDED	1331
CPSMSYSTEM	1332
CPSMVERSION	1333
CPSMAPI	1334
NOTSUPPORTED	1335
NOTVSNCONN	1336
INVALIDATTR	1337
APITASKERR	1338
CPSMSERVER	1339
APITASK	1340
PLEXMGR	1341
REQTIMEOUT	1342
AREATOOSMALL	1344
USRID	1345
VERSION	1348
FILTERMATCH	1352
INVALIDOBJ	1353
INVALIDVER	1354
INVALIDVERB	1356
NOSTORAGE	1357
NOSERVICE	1358
EXCEPTION	1359
INVALIDEVT	1360
DATAERROR	1361
CMAS	1362
FIRST	1363
NEXT	1364
EXPIRED	1365

Glossary

This glossary defines CICSPlex SM terms and abbreviations used in this book with other than their everyday meaning. Terms that are defined in the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994, are not defined here unless CICSPlex SM usage is different from the meaning given there.

If you cannot find the definition you need, refer to the *Dictionary of Computing* or the *CICSPlex SM Master Index*, SC33-1812.

A

action command. A CICSPlex SM command that affects one or more of the resources represented in a view. Action commands can be issued from either the COMMAND field in the control area of the information display panel or the line command field in a displayed view. Valid action commands are listed with the description of each view. See also *overtype field*.

action definition (ACTNDEF). In real-time analysis, a definition of the type of external notification that is to be issued when the conditions identified in an analysis definition are true.

activity. See *BTS activity*.

adjacent CMAS. A CICSPlex SM address space (CMAS) that is connected to the local CMAS via a direct CMAS-to-CMAS link. Contrast with *indirect CMAS*. See also *local CMAS*.

alter expression. A character string that defines the changes to be made to a resource attribute. An alter expression is made up of one or more attribute expressions.

alternate window. A window to which the results of a hyperlink can be directed. By default, the results of a hyperlink are displayed in the same window from which the hyperlink is initiated. Contrast with *current window*.

alternate window (ALT WIN) field. In the control area of an information display panel, the field in which you can specify an alternate window to receive the results of a hyperlink.

analysis definition. In real-time analysis, a definition of the evaluations to be performed on specified CICS resources, the intervals at which those evaluations are to be performed, and the actions to be taken when a notifiable condition occurs.

analysis group. In real-time analysis, a group of one or more analysis definitions, status definitions, or both. Analysis definitions and status definitions must belong to an analysis group if they are to be installed automatically in a CICS system when that system starts.

analysis point monitoring (APM). In real-time analysis, resource monitoring across multiple CICS systems within a CICSplex that results in a single notification of a condition, rather than one notification for each system. Contrast with *MAS resource monitoring*.

analysis point specification. In real-time analysis, a specification that identifies the CMASs that are to be responsible for analysis point monitoring.

analysis specification. In real-time analysis, a specification that establishes system availability monitoring or MAS resource monitoring within a group of CICS systems.

AOR. Application-owning region.

API. Application programming interface

APM. Analysis point monitoring.

application-owning region (AOR). In a CICSplex configuration, a CICS region devoted to running applications. For dynamic routing, the terms *requesting region*, *routing region*, and *target region* are used instead of AOR to signify the role of the region in the dynamic routing request.

ARM. Automatic restart manager.

ASU. Automatic screen update.

attribute. See *resource attribute, resource table attribute*.

attribute expression. A reference to a resource table attribute and, in some cases, its value. Attribute expressions are used to build filter expressions, modification expressions, and order expressions.

attribute value. The data currently associated with a resource table attribute. For example, the file attribute OPENSTATUS might have a value of CLOSED.

automatic restart manager (ARM). A recovery function of MVS/ESA 5.2 that provides improved availability for batch jobs and started tasks by restarting them automatically if they end unexpectedly. The affected batch job or started task can be restarted on the same system or on a different one, if the system itself has failed.

automatic screen update (ASU). A CICSPlex SM facility that automatically updates the data in all unlocked windows at user-defined intervals. See also *automatic screen update interval*.

automatic screen update interval. The time interval between one automatic screen update and the next. This interval can be set in the CICSPlex SM user profile or when the ASU facility is turned on. See also *automatic screen update (ASU)*.

B

BAS. Business Application Services

batched repository-update facility. A CICSPlex SM facility, invoked from the CICSPlex SM end user interface, for the bulk application of CICSPlex SM definitions to a CMAS data repository.

BTS. CICS business transaction services

BTS activity. One part of a process managed by CICS BTS. Typically, an activity is part of a *business transaction*.

BTS process. A collection of more than one CICS BTS *activities*. Typically, a process is an instance of a *business transaction*.

BTS set. See CICS system group

business application. Any set of CICS resources that represent a meaningful entity to an enterprise or a user (such as, Payroll).

Business Application Services (BAS). The component of CICSPlex SM that provides the ability to define and manage business applications in terms of their CICS resources and associated CICS systems. BAS provides a central definition repository for CICS systems, complete with installation facilities and the ability to restrict a CICSPlex SM request to those resources defined as being part of the business application. See also *business application, scope*.

business transaction. A self-contained business function, for example, the booking of an airline ticket.

C

CAS. Coordinating address space.

CBIPO. Custom-built installation process offering.

CBPDO. Custom-built product delivery offering.

CEDA. A CICS transaction that defines resources online. Using CEDA, you can update both the CICS system definition data set (CSD) and the running CICS system.

CICS Business Transaction Services (BTS). A CICS domain that supports an application programming interface (API) and services that simplify the development of *business transactions*.

CICS system. The entire collection of hardware and software required by CICS. In CICSPlex SM topology, a definition referring to a CICS system that is to be managed by CICSPlex SM. See also *CICSplex, CICS system group*.

CICS system group. A set of CICS systems within a CICSplex that can be managed as a single entity. In CICSPlex SM topology, the user-defined name, description, and content information for a CICS system group. A CICS system group can be made up of CICS systems or other CICS system groups. In CICS CICS business transaction services (BTS), a CBTS set, that is the set of CICS regions across which BTS processes and activities may execute. See also *CICSplex, CICS system*.

CICSplex. A CICS complex. A CICSplex consists of two or more CICS regions that are linked using CICS intercommunication facilities. The links can be either intersystem communication (ISC) or interregion communication (IRC) links, but within a CICSplex are more commonly IRC. Typically, a CICSplex has at least one terminal-owning region (TOR), more than one application-owning region (AOR), and may have one or more regions that own the resources being accessed by the AORs. In CICSPlex SM, a management domain. The largest set of CICS regions, or CICS systems, to be manipulated by CICSPlex SM as a single entity. CICS systems in a CICSplex being managed by CICSPlex SM do not need to be connected to each other. See also *CICS system, CICS system group*.

CICSPlex SM. IBM CICSPlex System Manager.

CICSPlex SM address space (CMAS). A CICSPlex SM component that is responsible for managing CICSplexes. A CMAS provides the single-system image for a CICSplex by serving as the interface to other CICSplexes and external programs. There must be at least one CMAS in each MVS image on which you are running CICSPlex SM. A single CMAS can manage CICS systems within one or more CICSplexes. See also *coordinating address space (CAS), managed application system (MAS)*.

CICSPlex SM token. Unique, 4-byte values that CICSPlex SM assigns to various elements in the API environment. Token values are used by CICSPlex SM to correlate the results of certain API operations with subsequent requests.

client program. In dynamic routing, the application program, running in the *requesting region*, that issues a remote link request.

CMAS. CICSPlex SM address space.

CMAS link. A communications link between one CICSPlex SM address space (CMAS) and another CMAS or a remote managed application system (remote MAS). CMAS links are defined when CICSPlex SM is configured.

CODB. A CICSPlex SM transaction for interactive, system-level debugging of CMASs and of CICS/ESA, CICS/MVS, and CICS/VSE MASs. CODB must be used only at the request of customer support personnel.

COD0. A CICSPlex SM transaction for interactive, method-level debugging of CMASs and of CICS/ESA, CICS/MVS, CICS/VSE, and CICS for OS/2 MASs. COD0 must be used only at the request of customer support personnel.

COLU. A CICSPlex SM transaction for generating reports about CMAS and local MAS components. COLU must be used only at the request of customer support personnel.

COMMAND field. In the control area of an information display panel, the field that accepts CICSPlex SM, ISPF, and TSO commands. Contrast with *option field*.

command-level interface. A CICSPlex SM API interface that uses the CICS translator to translate EXEC CPSM statements into an appropriate sequence of instructions in the source language.

Common Services. A component of CICSPlex SM that provides commonly requested services (such as GETMAIN, FREEMAIN, POST, and WAIT processing) to other CICSPlex SM components.

communication area (COMMAREA). A CICS area that is used to pass data between tasks that communicate with a given terminal. The area can also be used to pass data between programs within a task.

Communications. A component of CICSPlex SM that provides all services for implementing CMAS-to-CMAS and CMAS-to-MAS communication.

context. A named part of the CICSPlex SM environment that is currently being acted upon by CICSPlex SM. For configuration tasks, the context is a CICSPlex SM address space (CMAS); for all other tasks, it is a CICSplex. See also *scope*.

control area. The top three lines of an information display panel, containing the panel title, the screen update time, the short message area, the COMMAND and SCROLL fields, and the current window (CUR WIN) and alternate window (ALT WIN) fields.

coordinating address space (CAS). An MVS subsystem that provides ISPF end-user access to the CICSplex to be accessed. See also *CICSPlex SM address space, managed application system (MAS)*.

coordinating address space subsystem ID. Identifies the coordinating address space (CAS) which can be up to 4 characters, to be connected to when issuing CICSPlex SM requests. The name of the CAS is installation-dependent, and is defined in the CICSPlex SM user profile.

cross-system coupling facility (XCF). XCF is a component of MVS that provides functions to support cooperation between authorized programs running within a sysplex.

current window. The window to which the results of all commands issued in the COMMAND field are directed, unless otherwise requested. Contrast with *alternate window*.

current window (CUR WIN) field. In the control area of an information display panel, the field that contains the window number of the current window. You can change the number in this field to establish a new current window.

custom-built installation process offering (CBIPO). A product that simplifies the ordering, installation, and service of MVS system control programs and licensed programs by providing them with current updates and corrections to the software that is already integrated.

custom-built product delivery offering (CBPDO). A customized package of both products and service, or of service only, for MVS system control programs and licensed programs.

D

Data Cache Manager. A component of CICSPlex SM that manages logical cache storage for use by other CICSPlex SM components.

data repository. In CICSPlex SM, the VSAM data set that stores administrative data, such as topology and monitor definitions, for a CICSPlex SM address space (CMAS).

Data Repository. A component of CICSPlex SM that provides methods for creating, accessing, updating, and deleting data in the CICSPlex SM data repository. See also *Managed Object Services*.

Database Control (DBCTL). An IMS/ESA facility providing an interface between CICS/ESA and IMS/ESA that allows access to IMS DL/I full-function databases and to data-entry databases (DEDBs) from one or more CICS/ESA systems.

Database 2 (DB2). An IBM licensed program. DB2 is a full-function relational database management system that presents a data structure as a table consisting of a number of rows (or records) and a number of columns.

DBCTL. Database Control.

DB2. Database 2.

derived field. On a monitor view, a field whose value does not come directly from CICS or CICSPlex SM data, but is calculated based on the values in other fields. See also *derived value*.

derived value. A rate, average, or percentage that results from CICSPlex SM processing of CICS statistics.

display area. On an information display panel, the area where windows can be opened to display data. The display area appears below the control area. The bottom two lines of the display area can be used to display the PF key assignments in effect for a CICSPlex SM session.

display attributes. A CICSPlex SM user profile option that controls the appearance of the window information line, field headings, and threshold values in a view.

display command. A CICSPlex SM command that extends the ISPF interface to create and control a multiwindow environment.

distributed program link (DPL). Function of CICS intersystem communication that enables CICS to ship LINK requests between CICS regions.

distributed routing program (DSRTPGM). A CICS-supplied user-replaceable program that can be used to dynamically route:

CICS BTS processes and activities

DPL. Distributed program link.

DTR. Dynamic transaction routing.

dynamic routing. The automatic routing of a transaction or program, at the time it is initiated, from a requesting region to a suitable target region. Routing terminal data to an alternative transaction at the time the transaction is invoked. To do this, CICS allows the dynamic routing program to intercept the terminal data and redirect it to any system and transaction it chooses. See also dynamic routing program (EYU9XLOP)

dynamic routing program (EYU9XLOP). A user-replaceable CICS program that selects dynamically both the system to which a routing request is to be sent and the transaction's remote name. The alternative to using this program is to make these selections when a remote transaction is defined to CICS (static routing). See also *static routing*

dynamic transaction routing (DTR). The automatic routing of a transaction, at the time it is initiated, from a transaction-owning region (TOR) to a suitable application-owning region (AOR).

E

Environment Services System Services (ESSS). A component of CICSPlex SM that implements the formal MVS/ESA subsystem functions required by the product. ESSS provides cross-memory services, data space management, connection services, and lock management. An ESSS system address space is created at CICSPlex SM initialization and remains in the MVS image for the life of the IPL.

ESSS. Environment Services System Services.

evaluation definition. In real-time analysis, a definition of the resources that are to be sampled. When the result of an evaluation is true, an associated analysis definition is used to determine whether a notifiable condition has occurred.

event. A significant occurrence within the CICSplex or system for which the user has requested notification. For example, the end of processing, a subsystem failure, or any unusual condition in the system could be defined by a user as an event.

event notification. A CICSPlex SM notification of a significant occurrence within a CICSplex or CICS system.

extended diagnostic mode (XDM). A CICSPlex SM online internal diagnostic facility. XDM provides no information about resources managed by CICSPlex SM, and should be turned on only at the request of IBM customer support personnel. XDM can be turned on and off in the CICSPlex SM user profile.

external notification. In RTA, an event notification, generic alert, or operator message issued when a notifiable condition occurs.

F

file-owning region. In a CICSplex configuration, a CICS system devoted to managing CICS file access.

filter expression. A character string that consists of logical expressions to be used in filtering resource table records. A filter expression is made up of one or more attribute expressions.

FOR. File-owning region.

form. The way in which data obtained from a query is presented in a view. See also *query, view*.

G

generic alert. A Systems Network Architecture (SNA) Network Management Vector that enables a product to signal a problem to the network. CICSPlex SM uses generic alerts as part of its interface to NetView.

GMFHS. Graphic Monitor Facility host subsystem.

goal algorithm. In CICSPlex SM's workload balancing, an algorithm used to select an AOR to process a dynamic transaction. Using the goal algorithm, CICSPlex SM selects the AOR that is the least affected by conditions such as short-on-storage, SYSDUMP, and TRANDUMP; is the least likely to cause the transaction to abend; and is most likely to enable the transaction to meet response-time goals set for it using the Workload Manager component of MVS/ESA SP 5.1. Contrast with *queue algorithm*.

Graphic Monitor Facility host subsystem. A NetView feature that manages configuration and status updates for non-SNA resources.

H

hyperlink. A direct connection between the data in one CICSPlex SM view and a view containing related information. For example, from a view that lists multiple CICS resources, there may be a hyperlink to a detailed view for one of the resources. To use a hyperlink, place the cursor in the data portion of a hyperlink field and press Enter.

hyperlink field. On a CICSPlex SM view, a field for which a hyperlink is defined. The headings of hyperlink fields are shown in high intensity or color, depending on the terminal type.

I

IBM CICSPlex System Manager for MVS/ESA (CICSPlex SM). An IBM CICS system-management product that provides a single-system image and a single point of control for one or more CICSplexes that can be installed on heterogeneous operating systems.

indirect CMAS. A CICSPlex SM address space (CMAS) that the local CMAS can communicate with via an adjacent CMAS. There is no direct CMAS-to-CMAS link between the local CMAS and an indirect CMAS. Contrast with *adjacent CMAS*. See also *local CMAS*.

information display panel. The panel that supports the CICSPlex SM window environment. It consists of a control area and a display area. CICSPlex SM views are displayed in windows within the display area of this panel.

information display parameters. A CICSPlex SM user profile option that defines the initial screen configuration, how frequently the screen will be updated by ASU, and how long a window will wait for command processing to complete before timing out.

installation verification procedure (IVP). A procedure distributed with a system that tests the newly generated system to verify that the basic facilities of the system are functioning correctly.

interregion communication. Synonym for *multiregion operation*.

intersystem communication (ISC). Communication between separate systems by means of SNA networking facilities or by means of the application-to-application facilities of an SNA access method.

intertransaction affinity. A relationship between CICS transactions, usually the result of the ways in which information is passed between those transactions, that requires them to execute in the same CICS region. Intertransaction affinity imposes restrictions on the dynamic routing of transactions.

IRC. Interregion communication.

ISC. Intersystem communication.

IVP. Installation verification procedure.

K

Kernel Linkage. A component of CICSPlex SM that is responsible for building data structures and managing the interfaces between the other CICSPlex SM components. The environment built by Kernel Linkage is known as the method call environment.

L

line command field. In a CICSPlex SM view, the 3 character field, to the left of the data, that accepts action commands.

local CMAS. The CICSPlex SM address space (CMAS) that a user identifies as the current context when performing CMAS configuration tasks.

local MAS. A managed application system (MAS) that resides in the same MVS image as the CICSPlex SM address space (CMAS) that controls it and that uses the Environment Services System Services (ESSS) to communicate with the CMAS.

logical scope. A set of logically related CICS resources that are identified in a CICSPlex SM resource description. A logical scope can be used to qualify the context of a CICSPlex SM request.

M

maintenance point. A CICSPlex SM address space (CMAS) that is responsible for maintaining CICSPlex SM definitions in its data repository and distributing them to other CMASs involved in the management of a CICSplex. See also *data repository*.

Major object descriptor block (MODB). In CICSPlex SM, a control structure built by Kernel Linkage during initialization of a CICSPlex SM component that contains a directory of all methods that make up that component. The structure of the MODB is the same for all components.

Major object environment block (MOEB). In CICSPlex SM, a control structure built by Kernel Linkage during initialization of a CICSPlex SM component and pointed to by the MODB. The MOEB stores information critical to a CICSPlex SM component and anchors data used by the component. The structure of the MOEB is unique to the component it supports.

MAL. Message argument list.

managed application system (MAS). A CICS system that is being managed by CICSPlex SM. See *local MAS, remote MAS*.

managed object. A CICSPlex SM-managed CICS resource or a CICSPlex SM definition represented by a resource table. A view is based on a single managed object.

Managed Object Services. A subcomponent of the Data Repository component of CICSPlex SM that translates a request for data (from real-time analysis, for example) into the method calls required to obtain the data.

MAS. Managed application system.

MAS agent. A CICSPlex SM component that acts within a CICS system to provide monitoring and data collection for the CICSPlex SM address space (CMAS). The level of service provided by a MAS agent depends on the level of CICS the system is running under and whether it is a local or remote MAS. See also *CICSPlex SM address space (CMAS), local MAS, remote MAS*.

MAS resource monitoring (MRM). In real-time analysis, resource monitoring at the CICS system level; it results in one notification of a condition for each system in which it occurs. If the same condition occurs in three CICS systems where MAS resource monitoring is active, three notifications are issued. Contrast with *analysis point monitoring*.

Message argument list (MAL). In CICSPlex SM, a data structure passed between methods using Kernel Linkage method call services.

message line. On an information display panel, the line in the control area where a long message appears when the HELP command is issued in response to a short message. The message line temporarily overlays the CURR WIN and ALT WIN fields.

Message Services. A component of CICSPlex SM that provides services for building and issuing MVS/ESA console messages to other CICSPlex SM components.

meta-data. Internal data that describes the structure and characteristics of CICSPlex SM managed objects.

method. (Action.) An application programming interface (API) instruction that resolves into an EXEC CICS command, issued against one or more resources in one or more CICS systems, within the current context and scope.

method. In CICSPlex SM, one of the programs that make up a CICSPlex SM component. See also *message argument list (MAL)*.

mirror transaction. CICS transaction that recreates a request that is function shipped from one system to another, issues the request on the second system, and passes the acquired data back to the first system.

MODB. Major object descriptor block.

modification expression. A character string that defines the changes to be made to a resource attribute. A modification expression is made up of one or more attribute expressions.

MOEB. Major object environment block.

monitor definition. A user-defined statement of the specific resource occurrences (such as the program named PAYROLL) to be monitored by CICSPlex SM. A monitor definition can either be linked to a monitor specification as part of a monitor group or be installed directly into an active CICS system. See also *monitor group, monitor specification*.

monitor group. A user-defined set of CICSPlex SM monitor definitions that can either be linked to a monitor specification for automatic installation or be installed directly into an active CICS system. See also *monitor definition, monitor specification*.

monitor interval. The number of minutes that are to elapse before the statistics counters containing accumulated resource monitoring data are automatically reset. This value is part of a CICSplex definition and affects all of the CICS systems and CICS system groups associated with that CICSplex. See also *period definition, sample interval*.

monitor specification. A user-defined statement of the types of resources (such as programs) to be monitored by CICSPlex SM and how often data should be collected. A monitor specification is associated with a CICS system and is automatically installed each time the CICS system starts up. See also *monitor definition, monitor group*.

Monitoring Services. A component of CICSPlex SM that is responsible for monitoring resources within a CICS system and making the collected data available to other CICSPlex SM components.

MRM. MAS resource monitoring.

MRO. Multiregion operation.

MSM. MultiSystem Manager.

multiregion operation (MRO). Communication between CICS systems without the use of SNA network facilities. Synonymous with *interregion communication*.

MultiSystem Manager. An object-oriented, graphical systems management application that runs under NetView for MVS.

MVS image. A single instance of the MVS operating system.

MVS system. An MVS image together with its associated hardware.

N

NetView. An IBM network management product that can provide rapid notification of events and automated operations. CICSPlex SM can be set up to send generic alerts to NetView as part of its event processing capabilities.

NetView Graphic Monitor Facility (NGMF). A function of the NetView program that provides the network operator with a graphic topological presentation of a network controlled by the NetView program and that allows the operator to manage the network interactively.

NetView program. An IBM licensed program used to monitor and manage a network and to diagnose network problems.

NGMF. NetView Graphic Monitor Facility.

notification. A message that is generated asynchronously by a CICSPlex SM managed object to describe an event related to the object.

O

option field. On a CICSPlex SM menu, the field in which you can specify an option number or letter. Contrast with *command field*.

order expression. A character string that defines either the attributes to be used in sorting resource table records, or the attributes to be included in a resource table view. An order expression is made up of one or more attribute expressions.

override expression. A character string that defines the changes to be made to a resource attribute. An override expression is made up of one or more attribute expressions.

overtype field. On a CICSPlex SM view, a field containing a value that can be changed by typing a new value directly into the field. Values that can be overtyped are shown in high intensity or color, depending on the terminal type. Acceptable values for overtype fields are listed with the description of each view. See also *action command*.

P

parameter expression. A character string that defines the parameters required for an action to complete or a definition to be processed.

parameter repository. In CICSPlex SM, a data set that stores cross-system communication definitions that allow one coordinating address space (CAS) to communicate with other CASs.

period definition. A user-defined range of hours and minutes and the time zone to which that range applies. A period definition is used to indicate when an action, such as resource monitoring, is to occur. See also *monitor interval, sample interval*.

PlexManager. A service utility that can be used to manage the communication connections between multiple coordinating address spaces (CASs) and between a CAS and its associated CICSPlex SM address spaces (CMASs) and CICSplexes.

process. See *CICS BTS process*

processing thread. A connection between an application program and the CICSPlex SM API. A program can establish multiple processing threads, but each one is considered a unique API user; no resources can be shared across the boundary of a thread.

pseudoconversation. A CICS application designed to appear to the user as a continuous conversation, but that consists internally of multiple separate tasks.

Q

query. A request for specific data that is generated by a view command. See also *form, view*.

queue algorithm. In CICSPlex SM's workload balancing, an algorithm used to select an AOR to process a dynamic transaction. Using the queue algorithm, CICSPlex SM selects the AOR that has the shortest queue of transactions (normalized to MAXTASKs) waiting to be processed; is the least affected by conditions such as short-on-storage, SYSDUMP, and TRANDUMP; and is the least likely to cause the transaction to abend. Contrast with *goal algorithm*.

Queue Manager. A component of CICSPlex SM that creates and manages queues of data in a cache that is shared by a CMAS and its local MASs.

R

RACF. Resource Access Control Facility.

real-time analysis (RTA). A component of CICSPlex SM that is responsible for monitoring the status of a CICS system or resource against its desired status, and issuing one or more external notifications when deviations occur.

record pointer. An internal indicator of the next resource table record to be processed in a result set.

related scope. A CICS system where resources defined to CICSPlex SM as remote should be assigned and, optionally, installed as local resources. See also *target scope*.

remote MAS. A managed application system (MAS) that uses MRO or LU 6.2 to communicate with the CICSPlex SM address space (CMAS) that controls it. A remote MAS may or may not reside in the same MVS image as the CMAS that controls it.

requesting region. The region in which a dynamic routing request originates. For dynamic transaction routing and inbound client dynamic program link requests, this is typically a TOR; for dynamic START requests and peer-to-peer dynamic program link requests, this is typically an AOR.

resource. Any physical or logical item in a CICS system, such as a transient data queue, a buffer pool, a file, a program, or a transaction.

Resource Access Control Facility (RACF). An IBM licensed program that provides for access control by identifying and verifying the users to the system, authorizing access to protected resources, logging any detected unauthorized attempts to enter the system, and logging the detected accesses to protected resources.

resource assignment. A user-defined statement that selects resource definitions to be assigned to CICS systems and, optionally, specifies resource attributes to override those definitions. A resource assignment applies to a single resource type and must be associated with a resource description. See also *resource definition, resource description*.

resource attribute. A characteristic of a CICS resource, such as the size of a buffer pool.

resource definition. In CICSPlex SM, a user-defined statement of the physical and operational characteristics of a CICS resource. Resource definitions can be associated with resource descriptions as part of a resource group. See also *resource description, resource group*.

resource description. A user-defined set of CICSPlex SM resource definitions that can be automatically installed in CICS systems and named as a logical scope for CICSPlex SM requests. Resource descriptions represent the largest set of CICS resources that can be managed by CICSPlex SM as a single entity. A resource description can be associated with one or more resource assignments. See also *logical scope, resource assignment, resource definition*.

resource group. A user-defined set of CICSPlex SM resource definitions. A resource group can be associated with resource descriptions either directly or by means of resource assignments. See also *resource assignment, resource definition, resource description*.

Resource Object Data Manager (RODM). A component of the NetView program that operates as a cache manager and that supports automation applications. RODM provides an in-memory cache for maintaining real-time data in an address space that is accessible by multiple applications.

resource table. The external representation of a CICSPlex SM managed object. A resource table defines all the attributes, or characteristics, of a managed object.

resource table attribute. A characteristic of a CICSPlex SM managed object, as represented by a field in a resource table.

resource type. A group of related resources, such as files.

result set. A logical group of resource table records that can be accessed, reviewed, and manipulated by an API program.

retention period. For a monitored CICS system, the period of time for which monitor data is retained after the system becomes inactive. If a system is being monitored, becomes inactive, and remains inactive beyond the specified retention period, the monitor data is discarded. If the system becomes active before the retention period expires, the monitor data gathered before the system became inactive is retained, and monitoring continues.

RODM. Resource Object Data Manager.

routing region. The region in which the decision is made as to which is the most suitable target region for a dynamic routing request. For dynamic transaction routing, dynamic START requests, and inbound client dynamic program link requests, this is typcially a TOR; for dynamic peer-to-peer program link requests, this is typically an AOR.

RTA. real-time analysis.

run-time Interface. A CICSPlex SM API interface that accepts commands in the form of text strings and generates the appropriate API calls. The run-time interface supports programs written as REXX EXECs.

S

SAM. System availability monitoring.

sample interval. The duration, in seconds, between occurrences of data collection for a specific resource type. See also *monitor interval, period definition, resource type*.

scope. A named part of the CICSPlex SM environment that qualifies the context of a CICSPlex SM request. The scope can be the CICSplex itself, a CICS system, a CICS system group, or any set of CICS resources that are defined as a logical scope in a CICSPlex SM resource description. For configuration tasks, where the context is a CICSPlex SM address space (CMAS), the scope is ignored. When you are applying security, scope must be a single CICS system or CICSplex. It cannot be a CICS system group or any combination of individual CICSplexes or CICS systems. See also *context, logical scope*.

screen configuration. A user-defined, named layout of windows and the context, scope, view, and sort order associated with each. The initial configuration to be displayed when CICSPlex SM is accessed can be identified on the user profile.

screen repository. In CICSPlex SM, a data set that stores screen configuration definitions created by the SAVESCR display command. See also *screen configuration*.

selection list. In CICSPlex SM, a data set that stores cross-system communication definitions that allow one coordinating address space (CAS) to communicate with other CASs.

selection list. A list of named items, such as views or screen configurations, from which one can be selected.

server program. In dynamic routing, the application program specified on the link request, and which is executed in the *target region*.

service point. One of the combinations of products and contexts that is known to the coordinating address space (CAS) to which you are connected. See also *context*.

session control parameters. A CICSPlex SM user profile option that sets the coordinating address space (CAS) subsystem ID used for accessing CICSPlex SM views and controls the extended diagnostic mode (XDM).

short message area. In the control area of an information display panel, that part of the title line that displays short messages.

single point of control. The ability to access and manage all CICS systems and their resources in a CICSplex from a single terminal or user session.

single system image. The collection and presentation of data about multiple CICS systems as though they were a single CICS system. In CICSPlex SM, the single-system image is provided by the CICSPlex SM address space (CMAS).

specification. See *analysis specification, monitor specification, workload specification*.

Starter Set. A part of CICSPlex SM comprising sample CICSPlex SM definitions and sample JCL. The Starter Set samples may be used as supplied for educational purposes. They may also be copied and adapted for the customer environment.

static routing. Non-dynamic routing. The routing request is routed to a predetermined system. Static transaction routing occurs when NO is specified is the Dynamic field in either the transaction definition or the progam definition. In both cases, the request is routed to the system named in the Remote Sysid field.

status definition. In real-time analysis, a definition of a user-written program to be invoked at specified intervals to evaluate the status of a non-CICS resource.

summarized result set. A special type of result set that is produced by grouping, or summarizing, the resource table records in a result set. See also *result set*.

summary expression. A character string that consists of one or more summary options and the resource table attributes to which they apply. See also *summary option*.

summary option. A value that indicates how the attribute values in a resource table are to be summarized.

sysplex. A set of MVS systems communicating and cooperating with each other through specific multisystem hardware components and software services to process customer workloads.

system availability monitoring (SAM). In real-time analysis, the monitoring of CICS systems to determine whether: they are active during their defined hours of operation; they are experiencing a short-on-storage, SYSDUMP, TRANDUMP, MAXTASK, or STALL condition. If a CICS system becomes inactive or one of the specified conditions occurs, an external notification is issued.

system image. The representation of a program and its related data as it exists in main storage.

T

target region. The region selected from a set of target regions as the most suitable region in which to execute the work request. For all dynamic routing requests, this is typically an AOR.

target scope. A CICS system or CICS system group where resources defined to CICSPlex SM should be assigned and, optionally, installed. See also *related scope*.

temporary maintenance point. A CICSPlex SM address space (CMAS) that serves as the maintenance point when the identified maintenance point is unavailable. See also *maintenance point*.

terminal-owning region. In a CICSplex configuration, a CICS region devoted to managing the terminal network. For dynamic routing, the terms *requesting region* and *routing region* are used instead of TOR to signify the role of the region in the dynamic routing request.

thread. See *processing thread*.

time-period definition. A user-defined range of hours and minutes, and the time zone to which that range applies. A time-period definition is used to indicate when an action, such as resource monitoring, is to occur.

token. See *CICSPlex SM token, user token*.

topology. An inventory of CICS and CICSPlex SM resources, and a map of their relationships. CICSPlex SM supports the definition of resource and system topology.

topology definition. A named subset of CICS and CICSPlex SM resources. Topology definitions are user-created and can include CICSplexes, CICS systems, and CICS system groups.

Topology Services. A component of CICSPlex SM that is responsible for maintaining topology information about

CICSplexes and resources, and making it available to other CICSPlex SM components.

TOR. Terminal-owning region.

Trace Services. A component of CICSPlex SM that provides other CICSPlex SM components with the ability to write trace records to the CICS trace table and trace data sets. Trace Services also writes trace records created by a MAS to the trace table and data set of the managing CMAS.

transaction group. A user-defined, named set of transactions that determines the scope of workload balancing and the affinity relationships between transactions.

U

user token. Unique, 1- to 4-byte values that an API user can assign to asynchronous requests. User token values are not used by CICSPlex SM; they are simply held until the request is complete and then returned to the user.

V

view. In the CICSPlex SM API, a temporary, customized form of a resource table. A view can consist of some or all of the resource table attributes in any order. In the CICSPlex SM ISPF end-user interface, a formatted display of selected data about CICS resources or CICSPlex SM definitions. The data in a view is obtained from a query and can be presented in one or more forms. The data can be limited to a subset of CICSplex resources or definitions by establishing a context and scope.

view command. A CICSPlex SM command that displays a view in a window of the display area. The name of the view displayed matches the name of the view command. See also *view*.

W

window. In CICSPlex SM, a subdivision of the display area. The results of any CICSPlex SM view or display command are directed to a single window, which is the current window by default. Contrast with *view*. See also *current window*, *alternate window*.

window identifier. On a window information line, the field that identifies the window. A window identifier consists of a one-character status code and a number in the range 1 through 20.

window information line. The top line of each window in the display area. It includes the window identifier, the name of the view displayed in the window, the context and scope in effect, the date and time when the view was last refreshed, and the product name.

window number. A number assigned by CICSPlex SM to a window when it is opened. The window number is the second part of the window identifier on the window information line.

window status code. A one-character code that indicates whether a window is ready to receive commands, is busy processing commands, is not to be updated, or contains no data. It also indicates when an error has occurred in a window. The window status code is the first character of the window identifier on the window information line.

WLM. Workload Manager.

workload. The total number of transactions that a given CICSplex is intended to process in a specific period. For example, a workload could be expressed as a number of transactions per hour, or per day. In CICSPlex SM, a named set of transactions and CICS systems, acting as requesting regions, routing regions, and target regions that form a single, dynamic entity.

workload balancing. The technique of balancing a workload across multiple target regions that are capable of processing the work.

workload definition. A user-defined statement of the transaction groups associated with a CICS system that is an AOR. A workload definition can either be linked to a workload specification as part of a workload group or be installed directly into an active workload. See also *workload group, workload specification*.

workload group. A user-defined set of CICSPlex SM workload definitions that can either be linked to a workload specification for automatic installation or be installed directly into an active workload. See also *workload definition, workload specification*.

Workload Manager (WLM). A component of CICSPlex SM that is responsible for managing the transaction workload in a CICSplex through the use of dynamic transaction routing.

workload separation. The technique of separating a workload into discrete parts, and allocating specific transactions to specific AORs.

workload specification. A user-defined statement that identifies a workload and a set of CICS systems acting as AORs. A workload specification also provides default management criteria for transactions that are not defined to CICSPlex SM. It is associated with a CICS system that is a TOR and is automatically installed each time the CICS system starts up. See also *workload definition, workload group*.

X

XCF. Cross-system coupling facility of MVS/ESA.

XDM. Extended diagnostic mode

Index

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 request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to 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:

Information Development Department (MP095)
IBM United Kingdom Laboratories
Hursley Park
WINCHESTER,
Hampshire
SO21 2JN
United Kingdom

By fax:

– From outside the U.K., after your international access code use 44–1962–870229

– From within the U.K., use 01962–870229

Electronically, use the appropriate network ID:

– IBM Mail Exchange: GBIBM2Q9 at IBMMAIL

– IBMLink : HURSLEY(IDRCF)

– Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

The publication number and title

The topic to which your comment applies

Your name and address/telephone number/fax number/network ID.

IBM

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC34-5458-

Spine information:

IBM CICS TS for OS/390 CICSPlex SM Application Programming Reference Release 3