CICS® Transaction Server for z/OS™

CICSPlex® SM Application Programming Reference

Version 2 Release 1
Note!
Before using this information and the product it supports, be sure to read the general information under Notices on page 183.

This edition applies to Version 2 Release 1 of CICS Transaction Server for z/OS, program number 5697-E93, 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.

© Copyright International Business Machines Corporation 1995, 2001. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
# Contents

Preface ........................................ v
Who this book is for ............................. v
What you need to know ........................... v
How to use this book ............................. v
Notes on terminology ............................ v
CICS System Connectivity ....................... vi

Summary of changes ................................ ix

Chapter 1. Introduction to the commands .......... 1
Using the command-level interface ................. 1
  Command format ................................ 1
  Argument values .............................. 1
Using the run-time interface ..................... 5
  Command format ................................ 5
  Argument values .............................. 5
Syntax notation used in this book ............... 7
MVS/ESA restrictions ............................. 8
Language considerations .......................... 9
CICS and CICSPlex SM value data areas .......... 9
Length options .................................. 10
RESPONSE and REASON options .................... 10

Chapter 2. The API commands ..................... 13
ADDRESS .................................. 14
CANCEL .................................. 17
CONNECT .................................. 19
COPY .................................... 23
CREATE .................................. 27
DELETE ................................... 30
DISCARD .................................. 33
DISCONNECT ................................ 36
FETCH .................................... 38
FEEDBACK .................................. 42
GET ........................................ 48
GETDEF .................................... 53
GROUP .................................... 58
LISTEN .................................... 62
LOCATE .................................... 65
MARK ...................................... 70
ORDER ..................................... 74
PERFORM OBJECT ............................. 77
PERFORM SET ................................ 83
QUALIFY ................................... 88
QUERY ..................................... 91
RECEIVE ................................... 94
REFRESH .................................... 97
REMOVE .................................... 102
SET .......................................... 105
SPECIFY FILTER ............................... 111
SPECIFY VIEW ................................ 114
TERMINATE .................................. 117
TRANSLATE .................................. 118

© Copyright IBM Corp. 1995, 2001
Table of Contents

Chapter 3. REXX functions and commands .......................... 129
Functions ........................................................................ 129
  EYUAPI() ...................................................................... 130
  EYUNINIT() ..................................................................... 131
  EYUREAS() ...................................................................... 132
  EYURESP() ...................................................................... 133
  EYUTERM() ...................................................................... 134
Commands ........................................................................ 134
  TBUILD .......................................................................... 135
  TPARSE ........................................................................... 137

Appendix A. RESPONSE and REASON values ......................... 139

Appendix B. EYUDA values ..................................................... 147
EYUDA general values in numerical order .............................. 147
EYUDA general values in alphabetic order ............................. 158
EYUDA RESPONSE values in numerical order ................. 174
EYUDA RESPONSE values in alphabetic order .............. 175
EYUDA REASON values in numerical order ..................... 175
EYUDA REASON values in alphabetic order ............. 178

Bibliography ....................................................................... 181
CICS Transaction Server for z/OS ............................................ 181
  CICS books for CICS Transaction Server for z/OS .......... 181
  CICSPlex SM books for CICS Transaction Server for z/OS 182
  Other CICS books ......................................................... 182
Determining if a publication is current ................................ 182

Index .............................................................................. 185

Notices ............................................................................ 189
Sample programs ............................................................... 190
Programming interface information .................................... 190
Trademarks ............................................................... 190

Sending your comments to IBM ......................................... 191
Preface

This book provides programming information for the IBM CICSPlex® System Manager (CICSPlex SM) element of CICS® Transaction Server for z/OS®. 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 System Manager Concepts and Planning] book.

For guidance information on how to use the CICSPlex SM API see the [CICSPlex System Manager Application Programming Guide].

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

- [CICSPlex System Manager Managing Business Applications] For information about Business Application Services definitions.
- [CICSPlex System Manager Managing Resource Usage] For information about real-time analysis and Monitoring definitions.
- [CICSPlex System Manager 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 z/OS. The term **CICSp lex** (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:

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application programming interface</td>
</tr>
</tbody>
</table>
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 z/OS 2.1
- 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 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/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 z/OS 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>
<thead>
<tr>
<th>Table 1. Directly-connectable CICS systems by CICSPlex SM release</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICS system</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>CICS TS 2.1</td>
</tr>
<tr>
<td>CICS TS 1.3</td>
</tr>
<tr>
<td>CICS TS 1.2</td>
</tr>
<tr>
<td>CICS TS 1.1</td>
</tr>
<tr>
<td>CICS for MVS/ESA 4.1</td>
</tr>
<tr>
<td>CICS for MVS/ESA 3.3</td>
</tr>
<tr>
<td>CICS for MVS 2.1.2</td>
</tr>
<tr>
<td>CICS for OS/2 3.1</td>
</tr>
<tr>
<td>CICS system</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>CICS for OS/2 3.0</td>
</tr>
<tr>
<td>CICS/OS2 2.0.1</td>
</tr>
</tbody>
</table>
Summary of changes

This book is based on the Release 3 edition of the CICSPlex SM Application Programming Reference. The information in this book has been updated to incorporate changes made for CICSPlex SM for CICS Transaction Server for z/OS, Version 2 Release 1. Changes made since the last edition are indicated by vertical bars to the left of the change.
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(0140)
   CONTEXT(EYUPLX01) SCOPE(EYUCSG01)
   THREAD(THRDCTKN)
   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
using the command-level interface

- 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.
using the command-level interface

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.
using the command-level interface

\textit{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 — \texttt{FIXED BIN(15)}
- Fullword binary — \texttt{FIXED BIN(31)}
- Character string — \texttt{CHAR(n)} where “n” is the number of bytes.

If the data type is unspecified, \textit{data-ref} can refer to an element, array, or structure; for example, \texttt{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.

\textit{data-area}

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

\textit{ptr-ref}

Can be replaced by any PL/I reference of type \texttt{POINTER ALIGNED}.

\textit{cpsm-token}

Can be replaced by any named variable with a data type of fullword binary, \texttt{FIXED BIN(31)}.

\textbf{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 =\texttt{AL2(100)}, forms such as \texttt{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:

\textit{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(0140)' ,
'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:
using the run-time interface

**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
using the run-time interface

record into individual fields. For a description of this command, see "Chapter 3. REXX functions and commands" on page 129.

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:
## syntax notation used in this book

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>![A] ![B] ![C]</td>
<td>A set of mutually exclusive alternatives, one of which you <strong>must</strong> code.</td>
</tr>
<tr>
<td>![A] ![B] ![C]</td>
<td>A set of mutually exclusive alternatives, one of which you <strong>may</strong> code.</td>
</tr>
<tr>
<td>![A] ![B] ![C]</td>
<td>A set of alternatives, any number of which you may code.</td>
</tr>
<tr>
<td>![A]</td>
<td>Alternatives where A is the default.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>See the separate syntax fragment whose name is shown.</td>
</tr>
<tr>
<td><strong>Name:</strong> ![A] ![B]</td>
<td></td>
</tr>
</tbody>
</table>

**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 129.

2. In some CICS environments, the DFHVALUE function returns incompatible CVDA values for the following resource table attribute:

<table>
<thead>
<tr>
<th>Resource table</th>
<th>Attribute value</th>
<th>CICS Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCTRAN</td>
<td>RESSEC(RESSECSEX)</td>
<td>CICS/MVS</td>
</tr>
</tbody>
</table>

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:
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 118.

Note: For a list of the EYUDA values used by CICSPlex SM, see "Appendix B. EYUDA values" on page 147.

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 129.

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.
RESPONSE and REASON options

**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 System Manager Application Programming Guide. For a summary of RESPONSE and REASON values by command, see “Appendix A. RESPONSE and REASON values” on page 139.
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" on page 14
- "CANCEL" on page 17
- "CONNECT" on page 19
- "COPY" on page 23
- "CREATE" on page 27
- "DELETE" on page 30
- "DISCARD" on page 33
- "DISCONNECT" on page 36
- "FEEDBACK" on page 38
- "FETCH" on page 42
- "GET" on page 48
- "GETDEF" on page 53
- "GROUP" on page 58
- "LISTEN" on page 63
- "LOCATE" on page 65
- "MARK" on page 70
- "ORDER" on page 74
- "PERFORM OBJECT" on page 77
- "PERFORM SET" on page 83
- "QUALIFY" on page 88
- "QUERY" on page 91
- "RECEIVE" on page 94
- "REFRESH" on page 97
- "REMOVE" on page 102
- "SET" on page 105
- "SPECIFY FILTER" on page 111
- "SPECIFY VIEW" on page 114
- "TERMINATE" on page 117
- "TRANSLATE" on page 118
- "UNMARK" on page 121
- "UPDATE" on page 125
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.
CANCEL

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

Establish 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) — SIGNONPARAM(data-value)

CONTEXT(data-value) — SCOPE(data-value) — THREAD(data-ref)

RESPONSE(data-ref) — 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 System Manager 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
CONNECT

- A CICS system or CICS system group within the CICSpex
- A logical scope, as defined in a CICSpex 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 CICSpex. 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 CICSpex 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 CICSpex System Manager Application Programming Guide.

THREAD(data-ref)
Names a variable to receive the fullword token that CICSpex 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 CICSpex 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 CICSpex System Manager Application Programming Guide.

VERSION(data-value)
Identifies the release of CICSpex 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 CICSpex SM release, such as 0140 for CICS Transaction Server for z/OS, Version 2 Release 1.

Notes:
1. The VERSION value must be 0120 or greater. The API cannot access data from a release of CICSpex 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 \[CICSPlex System Manager Application Programming Guide\].

### 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.

```
COPY FROM(cpsm-token) TO(cpsm-token)
```

- `ALL`
- `CURRENT`
- `FILTER(cpsm-token)`
- `MARKED`
- `NOTFILTER(cpsm-token)`
- `NOTMARKED`

```
COUNT(data-ref)
```

- `REPLACE`
- `ADD`
- `UPDATE`

```
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 processing 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
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)
```

```
PARM(data-area)—PARMLEN(data-value)—CONTEXT(data-value)
```

```
THREAD(cpsm-token)—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 CICSpix 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 CICSpix.

**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 CICSpix.

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. See the [CICSPlex System Manager Resource Tables Reference](#) for a list of all permitted null values.

**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 System Manager 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 System Manager Application Programming Guide](#). For a
CREATE

description of the parameters that are valid for a given resource table, see
[CICSPlex System Manager 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.
  
  SOCRESOURCES
    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.

```
<table>
<thead>
<tr>
<th>DELETE</th>
<th>COUNT(data-ref)</th>
<th>RESULT(cpsm-token)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILTER(cpsm-token)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARKED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOTFILTER(cpsm-token)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOTMARKED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>THREAD(cpsm-token)</td>
<td>RESPONSE(data-ref)</td>
<td>REASON(data-ref)</td>
</tr>
</tbody>
</table>
```

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:
DELETE

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 a result set, filter, or 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.

Description

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

Related commands

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

Chapter 2. The API commands 33
DISCARD

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

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.
Retrieve diagnostic data.

```
FEEDBACK INTO(data-area) LENGTH(data-ref)
```

```
RESULT(cpsm-token) NEXT THREAD(cpsm-token)
FIRST COUNT(data-ref)
```

```
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 System Manager Resource Tables Reference](CICSPlex_System_Manager_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.
FEEDBACK

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.

VERSIONINVNL
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.

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.

- **APPLYSUMMARY**
  Apply any, some, or all of the following options to the summary records and retrieve the detail records associated with the summary records selected.
The APPLYSUMMARY option is only valid if the DETAIL option is also specified.

If the DETAIL option is specified without the APPLYSUMMARY option the result will be as described under the DETAIL option.

If neither the DETAIL option nor the APPLYSUMMARY option are specified but any combination of some or all of the following record selection options: MARKED, NOTMARKED, FILTER, and NOTFILTER are issued against a summary result set, the record selection options are applied to the summary result set and the selected summary records are retrieved.

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.
**FETCH**

**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.

You cannot explicitly position the record pointer in the source result set. When you specify DETAIL, the POSITION option refers to the record in the summary result set.

For more information on processing summarized result sets, see CICSPlex System Manager Application Programming Guide. For a description of the GROUP command, which creates summarized result sets, see "GROUP" on page 58.

**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.

NODATA The length is set to zero.

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
FETCH

- 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.

VERSIONINV
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

GET

Return a result set containing selected resource table records.

\[\text{GET} \rightarrow \text{OBJECT}(\text{data-value}) \rightarrow \text{COUNT}(\text{data-ref})\]

\[\rightarrow \text{CRITERIA}(\text{data-area}) \rightarrow \text{LENGTH}(\text{data-value}) \rightarrow \text{FILTER}(\text{cpsm-token})\]

\[\rightarrow \text{PARM}(\text{data-area}) \rightarrow \text{PARMLEN}(\text{data-value}) \rightarrow \text{NOWAIT} \rightarrow \text{TOKEN}(\text{data-value})\]

\[\rightarrow \text{CONTEXT}(\text{data-value}) \rightarrow \text{SCOPE}(\text{data-value}) \rightarrow \text{RESULT}(\text{cpsm-token})\]

\[\rightarrow \text{THREAD}(\text{cpsm-token}) \rightarrow \text{RESPONSE}(\text{data-ref}) \rightarrow \text{REASON}(\text{data-ref})\]

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:

<table>
<thead>
<tr>
<th>Resource table</th>
<th>Attribute value</th>
<th>CICS Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCTRAN</td>
<td>RESSEC(RESSECXT)</td>
<td>CICS/MVS</td>
</tr>
</tbody>
</table>

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\((\text{data-value})\)

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 System Manager 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 System Manager 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

- 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 CICSpex 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 CICSpex itself
- A CICS system or CICS system group within the CICSpex
- A logical scope, as defined in a CICSpex 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 CICSpex 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.

INVALIDDATA
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.
GET

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.

WORKLOAD
The workload identified on the API request is not available on the local CMAS.

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.

VERSIONINV
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.

\[
\begin{align*}
\text{GETDEF} & \quad \text{OBJECT}(data-value) \quad \text{ATTRIBUTE}(data-value) \\
\text{RESOURCE}(data-value) & \quad \text{COUNT}(data-ref) \quad \text{RESULT}(cpsm-token) \\
\text{THREAD}(cpsm-token) & \quad \text{RESPONSE}(data-ref) \quad \text{REASON}(data-ref)
\end{align*}
\]

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 54.

- 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

Add 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.
GETDEF

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.
GETDEF

**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).
GETDEF

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.

ENVIRONNERROR
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.

- GROUP — BY(data-value) — FROM(cpsm-token) — TO(cpsm-token)

- ALL
- FILTER(cpsm-token)
- MARKED
- NOTFILTER(cpsm-token)
- NOTMARKED
- COUNT(data-ref)
- SUMOPT(data-area) — LENGTH(data-value)
- THREAD(cpsm-token)
- 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.

- The GROUP command may be used only for attributes with a length less than 255. A FAILED EXCEPTION error occurs for attribute lengths of 255 or greater.

- For more information on processing summarized result sets, see [CICSPlex System Manager 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\textsuperscript{\textit{data-ref}}
\begin{itemize}
\item Names a variable to receive the number of resource table records in the target result set after this operation is complete.
\end{itemize}

FILTER\textsuperscript{\textit{cpsm-token}}
\begin{itemize}
\item Identifies a filter to be used for this operation. The FILTER option summarizes only those resource table records that meet the specified filter criteria.
\end{itemize}

The \textit{cpsm-token} value that identifies a filter is returned by the SPECIFY FILTER command.

FROM\textsuperscript{\textit{cpsm-token}}
\begin{itemize}
\item Identifies the source result set for this operation. The result set can be one produced by any of these commands:
\begin{itemize}
\item COPY
\item GET
\item PERFORM OBJECT.
\end{itemize}
\end{itemize}

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

LENGTH\textsuperscript{\textit{data-value}}
\begin{itemize}
\item A fullword value that specifies the length of the SUMOPT buffer.
\end{itemize}

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

MARKED
\begin{itemize}
\item 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.
\end{itemize}

NOTFILTER\textsuperscript{\textit{cpsm-token}}
\begin{itemize}
\item 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.
\end{itemize}

The \textit{cpsm-token} value that identifies a filter is returned by the SPECIFY FILTER command.

NOTMARKED
\begin{itemize}
\item 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.
\end{itemize}

REASON\textsuperscript{\textit{data-ref}}
\begin{itemize}
\item Names a variable to receive the fullword reason value returned by this command.
\end{itemize}

RESPONSE\textsuperscript{\textit{data-ref}}
\begin{itemize}
\item Names a variable to receive the fullword response value returned by this command.
\end{itemize}

SUMOPT\textsuperscript{\textit{data-area}}
\begin{itemize}
\item 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.
\end{itemize}
GROUP

For details on how to form a summary expression, see the CICSPlex System Manager Application Programming Guide. For a list of the default summary options for a given resource table, see the CICSPlex System Manager 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

LISTEN

Request a notification be sent to the processing thread.

Invocation

LISTEN EVENT(data-value) NOTIFICATION(data-ref)

FILTER(cpsm-token)
NOTFILTER(cpsm-token)

TOKEN(data-value)
CONTEXT(data-value)

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

Description

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

- 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 [CICSpexit System Manager 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 CICSpexit.

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 CICSpexit SM resource tables by type, see [CICSpexit System Manager 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 CICSpexit. For example, you could create a filter like this:

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

The \textit{cpsm-token} value that identifies a filter is returned by the SPECIFY FILTER command.

\texttt{NOTFILTER}(\textit{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 \textit{cpsm-token} value that identifies a filter is returned by the SPECIFY FILTER command.

\texttt{NOTIFICATION}(\textit{data-ref})

Names a variable to receive the fullword token that CICSpix SM assigns to this notification request.

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

\texttt{REASON}(\textit{data-ref})

Names a variable to receive the fullword reason value returned by this command.

\texttt{RESPONSE}(\textit{data-ref})

Names a variable to receive the fullword response value returned by this command.

\texttt{THREAD}(\textit{cpsm-token})

Identifies the API thread to be used for this operation. The \textit{cpsm-token} value that identifies a thread is returned by the CONNECT command.

\texttt{TOKEN}(\textit{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; CICSpix SM makes no use of the value. The token is returned by the RECEIVE command when an event of the specified type occurs.

\section*{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.

\textbf{OK} \hspace{1cm} The command completed processing successfully.

\textbf{ENVIRONERROR} \hspace{1cm} An environment error occurred for one of the following reasons:

\textbf{NOSERVICE} \hspace{1cm} The application stub program could not load the API service module.

\textbf{NOSTORAGE} \hspace{1cm} The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

\textbf{SOCRESRESOURCE} \hspace{1cm} A required resource that is owned by the CMAS is not available.

\textbf{FAILED} \hspace{1cm} The command failed for one of the following reasons:
LISTEN

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.

\[\text{LOCATE} \quad \text{pointer\_group} \quad \text{RESULT}(\text{cpsm\_token}) \quad \text{THREAD}(\text{cpsm\_token})\]

\[\text{RESPONSE}(\text{data\_ref}) \quad \text{REASON}(\text{data\_ref})\]

**pointer\_group**

- **POSITION\_group**
  - **TOP**
  - **BOTTOM**
  - **POSITION(data\_value)**
  - **FORWARD(data\_value)**
  - **BACKWARD(data\_value)**

- **FILTER\_group**
  - **FILTER(cpsm\_token)**
  - **FIRST**
  - **MARKED**
  - **NOTFILTER(cpsm\_token)**
  - **NOTMARKED**
  - **LAST**
  - **NEXT**
  - **PREV**

**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.
LOCATE

- 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.
LOCATE

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.
LOCATE

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.

```
MARK records_group | COUNT(data-ref)
```

```
INTO(data-area) LENGTH(data-ref) RESULT(cpsm-token)
```

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

**records_group**

- **CURRENT**
- **ALL**
- **FILTER(cpsm-token)**
- **NOTFILTER(cpsm-token)**
- **POSITION(data-value)**
- **PARM(data-area) PARMLEN(data-value)**

**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 System Manager 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.
MARK

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 System Manager 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
ORDER

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)
```

```
CRITERIA(data-area) LENGTH(data-value) FILTER(cpsm-token)
```

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

```
CONTEXT(data-value) SCOPE(data-value)
```

```
RESULT(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.
- The PERFORM OBJECT command contains two phases; the first is to build the result set, and the second is to take the requested action against the records in the result set. If an error occurs during the building of the result set, but the result set is not empty, the requested action will still be attempted on the records that are present in the result set.

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 System Manager 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.
PERFORM OBJECT

**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.


**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 System Manager Application Programming Guide](https://www.ibm.com/docs/en/cics-ts).

**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 System Manager Application Programming Guide](https://www.ibm.com/docs/en/cics-ts). For a description of the parameters that are required for a given resource table action, see the [CICSPlex System Manager Resource Tables Reference](https://www.ibm.com/docs/en/cics-ts).
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.
PERFORM OBJECT

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.

SOCRESsource  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.

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

ACTION
CONTEXT
CRITERIA
FILTER
LENGTH
OBJECT
PARM
Check the command description for valid parameter syntax.

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

- **API TASK**
  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.

- **WORKLOAD**
  The workload identified on the API request is not available on the local CMAS.

**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.
PERFORM OBJECT

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.

WARNING
The PERFORM OBJECT command may have only partially completed for one of the following reasons:

RESULT
During the building of the result set to be used on the command, a non-OK response was received. However some result set records were available and the requested action was successfully performed against them. Use the FEEDBACK command without the RESULT option to obtain information about the non-OK response.

ACTION
During the building of the result set to be used on the command, a non-OK response was received. However some result set records were available and the requested action was attempted. The action specified did not complete successfully on, at least one result set record due to a TABLEERROR or DATAERROR CICSPlex SM response or reason.

Use the FEEDBACK command without the RESULT option to obtain information about the error that occurred during the building of the result set. Use the FEEDBACK command with the RESULT option to obtain information about records that caused the TABLEERROR or DATAERROR response or reason.

WORKLOAD
The workload identified on the API request is not available on the local CMAS.
PERFORM SET

Performs an action on one or more resources.

```
PERFORM SET — ACTION(data-value)

- CURRENT
- FILTER(cpsm-token)
- MARKED
- NOTFILTER(cpsm-token)
- NOTMARKED
- POSITION(data-value)

- NOWAIT
- TOKEN(data-value)
- PARM(data-area) — PARMLEN(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 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 System Manager 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.
PERFORM SET

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 System Manager 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 System Manager Application Programming Guide. For a description of the parameters that are required for a given resource table action, see the CICSPlex System Manager 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.

SOCRESOURCEx
A required resource that is owned by the CMAS is not available.

FAILED
The command failed for one of the following reasons:
PERFORM SET

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 CICSpinx and no other CMAS is available that does manage the CICSpinx.

SCOPE Either none of the MASs in the specified scope are available or none of them support the requested action.

WORKLOAD The workload identified on the API request is not available on the local CMAS.

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

QUALIFY

Define the CICSPlex SM context and scope.

- QUALIFY-CONTEXT(data-value) - THREAD(cpsm-token)
- SCOPE(data-value)

RESPONSE(data-ref) - 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:

NO SERVICE
The application stub program could not load the API service module.

NO STORAGE
The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

SOC RESOURCE
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.

INVALID PARM 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.

NOT AVAILABLE 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.

PLEX MGR The CMAS to which the processing thread is currently connected does not participate in managing the specified CICSp lex and no other CMAS is available that does manage the CICSp lex.
QUALIFY

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

VERSIONINV
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.

```plaintext
QUERY RESULT(cpsm-token) COUNT(data-ref) DATALENGTH(data-ref)
```

```plaintext
OBJECT(data-ref) TYPE(data-ref) CONTEXT(data-ref)
```

```plaintext
SCOPE(data-ref) THREAD(cpsm-token) RESPONSE(data-ref)
```

```plaintext
REASON(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.
QUERY

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.

SCOPE(data-ref)
Names a variable to receive the scope associated with the result set. This value may be blank for result sets containing CMAS type resources.

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:
NO SERVICE The application stub program could not load the API service module.

NO STORAGE The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

SOC RESOURCE 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.
QUERY

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.

\[\text{RECEIVE—INTO} (\text{data-area}) — \text{LENGTH} (\text{data-ref}) — \text{OBJECT} (\text{data-ref})\]

\[\text{THREAD} (\text{cpsm-token})\]

\[\text{WAIT}\]

\[\text{TOKEN} (\text{data-ref}) — \text{DELAY} (\text{data-value}) — \text{IMMEDIATE}\]

\[\text{RESPONSE} (\text{data-ref}) — \text{REASON} (\text{data-ref})\]

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 System Manager 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.
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.

SOCRESOURCENot available
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.

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 System Manager 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.

```plaintext
REMOVE—OBJECT(data-value)—FROM(data-area)—LENGTH(data-value)
```

- **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 System Manager 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 System Manager Application Programming Guide](#). For a description of the parameters that are valid for a given resource table, see the [CICSPlex System Manager Resource Tables Reference](#).

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

### Description

This command 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
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.

- SOCRESOURCES
  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.
REMOVE

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.

NOTVSNCNN
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.

![Command Diagram]

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.
SET

**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.


**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 \textit{cpsm-token} value that identifies a filter is returned by the \texttt{SPECIFY FILTER} command.

\textbf{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 \texttt{MARK} and \texttt{UNMARK} commands.

\textbf{NOWAIT}  
Returns control to your program as soon as the \texttt{SET} command has been accepted, which allows the command to be processed asynchronously.

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

Note: If you specify the \texttt{TOKEN} option, the \texttt{NOWAIT} option is assumed by default.

\textbf{POSITION}(\textit{data-value})  
Modifies the \textit{n}th resource table record in the result set. When you specify \texttt{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 \texttt{POSITION(5)}.

\textbf{REASON}(\textit{data-ref})  
Names a variable to receive the fullword reason value returned by this command.

\textbf{RESPONSE}(\textit{data-ref})  
Names a variable to receive the fullword response value returned by this command.

\textbf{RESULT}(\textit{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:

- \texttt{COPY}
- \texttt{GET}
- \texttt{GROUP}
- \texttt{PERFORM OBJECT}.  

\section*{Chapter 2. The API commands}
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.

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.
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 assigns 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 System Manager 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.
**SPECIFY FILTER**

**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.

**INVALIDDATA**
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 System Manager Application Programming Guide](https://www.ibm.com). You should check that any new resource tables do not have the same names as any customized views. For more details, see [CICSPlex System Manager Application Programming Guide](https://www.ibm.com).

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 System Manager Application Programming Guide](https://www.ibm.com).

**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 System Manager Application Programming Guide](https://www.ibm.com).
**SPECIFY VIEW**

**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:
SPECIFY VIEW

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.
## 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 123.

- In some CICS environments, the DFHVALUE function returns incompatible CVDA values for the following resource table attribute:

<table>
<thead>
<tr>
<th>Resource table</th>
<th>Attribute value</th>
<th>CICS Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCTRAN</td>
<td>RESSEC(RESSECEXT)</td>
<td>CICS/MVS</td>
</tr>
</tbody>
</table>

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.

**ENVIROMEERROR**
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.

**SOCRESOURSE**
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.
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.

**VERSIONINV**
- 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.

\[ \text{UNMARK} \text{ records\_group } \text{COUNT}(\text{data-ref}) \]
\[ \text{INTO}(\text{data-area}) \text{LENGTH}(\text{data-ref}) \]
\[ \text{RESULT}(\text{cpsm-token}) \]
\[ \text{THREAD}(\text{cpsm-token}) \text{RESPONSE}(\text{data-ref}) \text{REASON}(\text{data-ref}) \]

records\_group

- \text{CURRENT}
- \text{ALL}
- \text{FILTER}(\text{cpsm-token})
- \text{NOTFILTER}(\text{cpsm-token})
- \text{POSITION}(\text{data-value})
- \text{PARM}(\text{data-area}) \text{PARMLEN}(\text{data-value})

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

- \text{ALL}
  Removes the marks from all resource table records in the result set.
- \text{COUNT}(\text{data-ref})
  Names a variable to receive the number of resource table records that could not be unmarked.
- \text{CURRENT}
  Removes the mark from only the current resource table record.
- \text{FILTER}(\text{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 \text{cpsm-token} value that identifies a filter is returned by the SPECIFY FILTER command.
- \text{INTO}(\text{data-area})
  Identifies a buffer to receive a list of resource table records that could not be unmarked.
UNMARK

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 System Manager 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.
UNMARK

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.
  NOTVSNCN
    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)  CONTEXT(data-value)  THREAD(cpsm-token)
```

```
RESPONSE(data-ref)  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 CICSpex 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 CICSpex.

**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 CICSpex.

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.
UPDATE

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 System Manager 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 System Manager 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 System Manager Application Programming Guide. For a description of the parameters that are valid for a given resource table, see the CICSPlex System Manager 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.
The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

One of the CMASs or MASs to which the request was directed did not respond.

A required resource that is owned by the CMAS is not available.

The command failed for one of the following reasons:

Command processing abended.

Command processing encountered an exceptional condition.

An invalid data error occurred for one of the following reasons:

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

An attribute listed in the MODIFY buffer is not modifiable.

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

Check the command description for valid parameter syntax.

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

The API control subtask is not active.

A CMAS to which the request was directed is not available.

The CMAS to which the processing thread is connected is not available for API processing.

The maintenance point for the current context is not available.

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

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 CICSpix SM. These functions and commands can be used only with the REXX run-time interface.

The REXX functions supplied with CICSpix 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 CICSpix System Manager 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()" on page 130
- "EYUINIT()" on page 131
- "EYUREAS()" on page 132
- "EYURESP()" on page 133
- "EYUTERM()" on page 134

Functions
**EYUAPI()**

**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.

\[ \text{var} = \text{EYUINIT()} \]

**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 (\( \text{var} \)).

- \( 0 \) The EYUINIT function was successful.
- \( 1 \) The EYUINIT function failed.
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.
Terminate the CICSPlex SM API environment and release any allocated resources.

```rexx
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).

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The EYUTERM function was successful.</td>
</tr>
<tr>
<td>1</td>
<td>The EYUTERM function failed.</td>
</tr>
</tbody>
</table>

### 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 135
- TPARSE on page 137
Build a resource table record from a set of variables.

\[\text{TBUILD—OBJECT}(\text{data-value})—\text{PREFIX}(\text{data-value})—\text{STATUS}(\text{data-ref})\]

\[\text{VAR}(\text{data-area})—\text{THREAD}(\text{cpsm-token})\]

**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 System Manager Application Programming Guide](#). For a complete description of a particular resource table and its attributes, see the [CICSPlex System Manager 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.
TBUILD

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 System Manager 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.

\[ \text{TPARSE} \text{ OBJECT(data-value) \text{ PREFIX(data-value) \text{ STATUS(data-ref)}} \]

\[ \text{VAR(data-area)} \text{ ASIS \text{ THREAD(cpsm-token)}} \]

**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 System Manager 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.
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 System Manager 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 147](#). For a discussion of the RESPONSE and REASON options, see [CICSPlex System Manager Application Programming Guide](#).

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>RESPONSE</th>
<th>REASONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>OK</td>
<td>ENVIRONERROR, NOSERVICE, NOSTORAGE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>ECB, SENTINEL, THREAD</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td>CANCEL</td>
<td>OK</td>
<td>ENVIRONERROR, NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>NOTIFICATION, THREAD</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td>CONNECT</td>
<td>OK</td>
<td>ENVIRONERROR, APITASKERR, NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>CONTEXT, SCOPE, SIGNONPARM, USRID, VERSION</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI, CPSMSERVER, CPSMSYSTEM, CPSMVERSION</td>
</tr>
<tr>
<td></td>
<td>NOTPERMIT</td>
<td>USRID</td>
</tr>
<tr>
<td></td>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED</td>
</tr>
<tr>
<td>COPY</td>
<td>OK</td>
<td>NODATA</td>
</tr>
<tr>
<td></td>
<td>BUSY</td>
<td>FROM, TO</td>
</tr>
<tr>
<td></td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCES</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INCOMPATIBLE</td>
<td>INVALIDOBJ</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>FILTER, FROM, NOTFILTER, THREAD, TO</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td>CREATE</td>
<td>OK</td>
<td>ENVIRONERROR, NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCES</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1995, 2001
**RESPONSE and REASON values**

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>RESPONSE</th>
<th>REASONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALIDPARM</td>
<td>CONTEXT, FROM, LENGTH, OBJECT, PARM, PARMLEN, THREAD</td>
<td></td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CMAS, CPSMAPI, MAINTPOINT</td>
<td></td>
</tr>
<tr>
<td>NOTPERMIT</td>
<td>USRID</td>
<td></td>
</tr>
<tr>
<td>SERVERGONE</td>
<td>TABLEERROR, DATAERROR, INVALIDATTR, INVALIDVER</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
<td></td>
</tr>
</tbody>
</table>

**DELETE**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>NODATA</td>
<td></td>
</tr>
<tr>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCES, SOLRESOURCES</td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>FILTER, NOTFILTER, RESULT, THREAD</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

**DISCARD**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCES</td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td>INUSE</td>
<td>FILTER, VIEW</td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>FILTER, RESULT, THREAD, VIEW</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

**DISCONNECT**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCES</td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>THREAD</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

**FEEDBACK**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>NODATA</td>
<td>AREATOOSMALL</td>
</tr>
<tr>
<td>WARNING</td>
<td></td>
</tr>
<tr>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE</td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>COUNT, INTO, LENGTH, RESULT, THREAD</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

**FETCH**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>NODATA</td>
<td>BACKWARD, FORWARD</td>
</tr>
<tr>
<td>WARNING</td>
<td>AREATOOSMALL</td>
</tr>
<tr>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCES, SOLRESOURCES</td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
</tbody>
</table>
## RESPONSE and REASON values

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>RESPONSE</th>
<th>REASONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALIDPARM</td>
<td>COUNT, FILTER, INTO, LENGTH, NOTFILTER, POSITION, RESULT, THREAD</td>
<td></td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
<td></td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>OK, SCHEDULED, NODATA, BUSY, ENVIRONERROR, FAILED, INVALIDDATA, INCOMPATIBLE, INVALIDPARM, NOTAVAILABLE, NOTFOUND, NOTPERMIT, SERVERGONE, TABLEERROR, VERSIONINVL</td>
<td></td>
</tr>
<tr>
<td>GETDEF</td>
<td>OK, NODATA, BUSY, ENVIRONERROR, FAILED, INCOMPATIBLE, INVALIDPARM, NOTAVAILABLE, SERVERGONE, TABLEERROR, VERSIONINVL</td>
<td></td>
</tr>
<tr>
<td>GROUP</td>
<td>OK, NODATA, BUSY, ENVIRONERROR, FAILED, INCOMPATIBLE, INVALIDPARM, NOTAVAILABLE, SERVERGONE, VERSIONINVL</td>
<td></td>
</tr>
<tr>
<td>LISTEN</td>
<td>OK, ENVIRONERROR, FAILED, INCOMPATIBLE, INVALIDPARM, NOTAVAILABLE, SERVERGONE</td>
<td></td>
</tr>
</tbody>
</table>

Appendix A. RESPONSE and REASON values | 141
### RESPONSE and REASON values

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>RESPONSE</th>
<th>REASONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCATE</td>
<td>VERSIONINVNL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td>OK</td>
<td>NODATA</td>
<td>BACKWARD, FORWARD</td>
</tr>
<tr>
<td>BUSY</td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE, SOLRESOURCE</td>
</tr>
<tr>
<td>FAILED</td>
<td>INVALIDPARM</td>
<td>BACKWARD, FILTER, FORWARD, NOTFILTER, POSITION, RESULT, THREAD</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>VERSIONINVNL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td>MARK</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>NODATA</td>
<td>WARNING</td>
<td>AREATOOSMALL, DATAERROR</td>
</tr>
<tr>
<td>BUSY</td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE, SOLRESOURCE</td>
</tr>
<tr>
<td>FAILED</td>
<td>INVALIDPARM</td>
<td>COUNT, FILTER, INTO, LENGTH, NOTFILTER, PARM, PARMLEN, RESULT, THREAD</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>SERVERGONE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td>SERVERGONE</td>
<td>VERSIONINVNL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td>ORDER</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>BUSY</td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE SOCRESOURCE</td>
</tr>
<tr>
<td>FAILED</td>
<td>INVALIDPARM</td>
<td>BY, LENGTH, RESULT, THREAD</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>SERVERGONE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td>SERVERGONE</td>
<td>VERSIONINVNL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td>PERFORM</td>
<td>OBJECT</td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td>SCHEDULED</td>
<td>BACKWARD, FORWARD</td>
</tr>
<tr>
<td>NODATA</td>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, REQTIMELIMIT, SOCRESOURCE</td>
<td></td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
<td></td>
</tr>
<tr>
<td>INVALIDDATA</td>
<td>PARM, CRITERIA</td>
<td>ACTION, CONTEXT, CRITERIA, FILTER, LENGTH, OBJECT, PARM, PARMLEN, RESULT, SCOPE, THREAD, TOKEN</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CMSAPL, MAINTPOINT, PLEXMGR, SCOPE, WORKLOAD</td>
<td></td>
</tr>
<tr>
<td>NOTFOUND</td>
<td>Action, ATTRIBUTE</td>
<td></td>
</tr>
<tr>
<td>NOTPERMIT</td>
<td>USRID</td>
<td></td>
</tr>
<tr>
<td>SERVERGONE</td>
<td>DATAERROR</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVNL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
<td></td>
</tr>
<tr>
<td>WARNING</td>
<td>RESULT, ACTION</td>
<td></td>
</tr>
</tbody>
</table>
### RESPONSE and REASON values

<table>
<thead>
<tr>
<th>COMMAND RESPONSE</th>
<th>REASONS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PERFORM SET</strong></td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>SCHEDULED</td>
<td>BACKWARD, FORWARD</td>
</tr>
<tr>
<td>NODATA</td>
<td>RESULT</td>
</tr>
<tr>
<td>BUSY</td>
<td></td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, REQTIMEOUT</td>
</tr>
<tr>
<td></td>
<td>SOCRESOURCE</td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td>INVALIDATA</td>
<td>PARM, CRITERIA</td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>ACTION, FILTER, NOTFILTER, PARM, PARMLEN, POSITION, RESULT, THREAD, TOKEN</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CMAS, CPSMAPI, MAINTPOINT, PLEXMGR, SCOPE, WORKLOAD</td>
</tr>
<tr>
<td>NOTFOUND</td>
<td>ACTION, ATTRIBUTE</td>
</tr>
<tr>
<td>NOTPERMIT</td>
<td>USRID</td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td>TABLEERROR</td>
<td>DATAERROR</td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td><strong>QUALIFY</strong></td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>CONTEXT, SCOPE, THREAD</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI, PLEXMGR</td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td><strong>QUERY</strong></td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>CONTEXT, OBJECT, RESULT, THREAD, TYPE</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td><strong>RECEIVE</strong></td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>NODATA</td>
<td></td>
</tr>
<tr>
<td>WARNING</td>
<td>AREATOOSMALL</td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>DELAY, INTO, LENGTH, OBJECT, THREAD, TOKEN</td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td><strong>REFRESH</strong></td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>SCHEDULED</td>
<td>RESULT</td>
</tr>
<tr>
<td>BUSY</td>
<td></td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE</td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>COUNT, FILTER, NOTFILTER, RESULT, THREAD, TOKEN</td>
</tr>
</tbody>
</table>
## RESPONSE and REASON values

<table>
<thead>
<tr>
<th>COMMAND RESPONSE</th>
<th>REASONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CMAS, CPSMAPI, MAINTPOINT, SCOPE</td>
</tr>
<tr>
<td>NOTPERMIT</td>
<td>USRID</td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td>TABLEERROR</td>
<td>DATAERROR</td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

### REMOVE

<table>
<thead>
<tr>
<th>OK</th>
<th>ENVIRONERROR</th>
<th>FAILED</th>
<th>INVALIDPARAM</th>
<th>NOTAVAILABLE</th>
<th>NOTPERMIT</th>
<th>SERVERGONE</th>
<th>TABLEERROR</th>
<th>VERSIONINVL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE</td>
<td>ABENDED, EXCEPTION</td>
<td>CONTEXT, FROM, LENGTH, OBJECT, PARM, PARMLEN, THREAD</td>
<td>APITASK, CMAS, CPSMAPI, MAINTPOINT</td>
<td>USRID</td>
<td></td>
<td>DATAERROR, INVALIDATTR, INVALIDVER</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

### SET

<table>
<thead>
<tr>
<th>OK</th>
<th>SCHEDULED</th>
<th>BUSY</th>
<th>ENVIRONERROR</th>
<th>FAILED</th>
<th>INVALIDDATA</th>
<th>INVALIDPARAM</th>
<th>NOTAVAILABLE</th>
<th>NOTFOUND</th>
<th>NOTPERMIT</th>
<th>SERVERGONE</th>
<th>TABLEERROR</th>
<th>VERSIONINVL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE</td>
<td>ABENDED, EXCEPTION</td>
<td>MODIFY, NOTSUPPORTED</td>
<td>ATTRIBUTE, COUNT, FILTER, LENGTH, MODIFY, NOTFILTER, POSITION, RESULT, THREAD, TOKEN</td>
<td>APITASK, CMAS, CPSMAPI, MAINTPOINT, SCOPE</td>
<td>ACTION, ATTRIBUTE</td>
<td>USRID</td>
<td></td>
<td>DATAERROR, INVALIDVER</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

### SPECIFY FILTER

<table>
<thead>
<tr>
<th>OK</th>
<th>SPECIFY VIEW</th>
<th>DUPE</th>
<th>ENVIRONERROR</th>
<th>FAILED</th>
<th>INVALIDDATA</th>
<th>INVALIDPARAM</th>
<th>NOTAVAILABLE</th>
<th>NOTFOUND</th>
<th>SERVERGONE</th>
<th>VIEWERROR</th>
<th>VERSIONINVL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
<td>ABENDED, EXCEPTION</td>
<td>CRITERIA</td>
<td>CRITERIA, FILTER, LENGTH, OBJECT, THREAD</td>
<td>APITASK, CPSMAPI</td>
<td>ATTRIBUTE</td>
<td></td>
<td>VIEW</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

### SPECIFY VIEW

<table>
<thead>
<tr>
<th>OK</th>
<th></th>
<th>_DUPE</th>
<th>ENVIRONERROR</th>
<th>FAILED</th>
<th>INVALIDDATA</th>
<th>INVALIDPARAM</th>
<th>NOTAVAILABLE</th>
<th>NOTFOUND</th>
<th>SERVERGONE</th>
<th>TABLEERROR</th>
<th>VERSIONINVL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
<td>ABENDED, EXCEPTION</td>
<td>CRITERIA</td>
<td>ATTRIBUTE, FIELDS, LENGTH, OBJECT, THREAD, VIEW</td>
<td>APITASK, CPSMAPI</td>
<td>ATTRIBUTE</td>
<td></td>
<td>DATAERROR, INVALIDVER</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>
## RESPONSE and REASON values

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>RESPONSE</th>
<th>REASONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TERMINATE</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENED, EXCEPTION</td>
</tr>
<tr>
<td>TRANSLATE</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>ATTRIBUTE, FROMCHAR, FROMCV, OBJECT, THREAD, TOCHAR, TOCV</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TABLEERROR</td>
<td>INVALIDVER, INVALIDVER</td>
</tr>
<tr>
<td></td>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td>UNMARK</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NODATA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WARNING</td>
<td>AREATOOSMALL, DATAERROR</td>
</tr>
<tr>
<td></td>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td></td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE, SOLRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>COUNT, FILTER, INTO, LENGTH, NOTFILTER, PARM, PARMLEN, RESULT, THREAD</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
<tr>
<td>UPDATE</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>CONTEXT, FROM, LENGTH, MODIFY, OBJECT, PARM, PARMLEN, RESULT, THREAD</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CMA, CPSMAPI, MAINTPOINT</td>
</tr>
<tr>
<td></td>
<td>NOTPERMIT</td>
<td>USRID</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TABLEERROR</td>
<td>DATAERROR, INVALIDATTR, INVALIDVER</td>
</tr>
<tr>
<td></td>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>
RESPONSE and REASON values
Appendix B. EYUDA values

This appendix lists the CICSPlex SM API EYUDA values and their numeric equivalents. There are three types of EYUDAs:

**General**
Values that CICSPlex SM uses to describe or define a resource. These EYUDAs have numeric values in the range of 1–429. See [EYUDA general values in numerical order](#).

**RESPONSE**
Values returned by the RESPONSE option of an API command. These EYUDAs have numeric values in the range of 1024–1042. See [EYUDA RESPONSE values in numerical order](#) on page 174.

**REASON**
Values returned by the REASON option of an API command. These EYUDAs have numeric values in the range of 1280–1366. See [EYUDA REASON values in numerical order](#) on page 175.

EYUDA information is provided in the following sections:
- [EYUDA general values in numerical order](#)
- [EYUDA general values in alphabetic order](#) on page 158
- [EYUDA RESPONSE values in numerical order](#) on page 174
- [EYUDA RESPONSE values in alphabetic order](#) on page 175
- [EYUDA REASON values in numerical order](#) on page 175

---

**EYUDA general values in numerical order**

This section lists the general EYUDAs in numerical order and shows the character value for each.

<table>
<thead>
<tr>
<th>Value</th>
<th>EYUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N_A</td>
</tr>
<tr>
<td>0</td>
<td>NA</td>
</tr>
<tr>
<td>1</td>
<td>YES</td>
</tr>
<tr>
<td>2</td>
<td>NO</td>
</tr>
<tr>
<td>3</td>
<td>ON</td>
</tr>
<tr>
<td>4</td>
<td>OFF</td>
</tr>
<tr>
<td>5</td>
<td>VALID</td>
</tr>
<tr>
<td>6</td>
<td>INVALID</td>
</tr>
<tr>
<td>7</td>
<td>TRUE</td>
</tr>
<tr>
<td>8</td>
<td>FALSE</td>
</tr>
<tr>
<td>9</td>
<td>VLS</td>
</tr>
<tr>
<td>10</td>
<td>LS</td>
</tr>
<tr>
<td>11</td>
<td>LW</td>
</tr>
<tr>
<td>12</td>
<td>NM</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>-------------------</td>
</tr>
<tr>
<td>13</td>
<td>HW</td>
</tr>
<tr>
<td>14</td>
<td>HS</td>
</tr>
<tr>
<td>15</td>
<td>VHS</td>
</tr>
<tr>
<td>16</td>
<td>EQ</td>
</tr>
<tr>
<td>17</td>
<td>NE</td>
</tr>
<tr>
<td>18</td>
<td>GT</td>
</tr>
<tr>
<td>19</td>
<td>LT</td>
</tr>
<tr>
<td>20</td>
<td>LOW</td>
</tr>
<tr>
<td>21</td>
<td>HIGH</td>
</tr>
<tr>
<td>22</td>
<td>NORMAL</td>
</tr>
<tr>
<td>23</td>
<td>IMMEDIATE</td>
</tr>
<tr>
<td>24</td>
<td>TAKEOVER</td>
</tr>
<tr>
<td>25</td>
<td>SHUT</td>
</tr>
<tr>
<td>26</td>
<td>NOSHUT</td>
</tr>
<tr>
<td>27</td>
<td>GLOBAL</td>
</tr>
<tr>
<td>28</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>29</td>
<td>SUSPEND</td>
</tr>
<tr>
<td>30</td>
<td>VALUE</td>
</tr>
<tr>
<td>31</td>
<td>THRESHOLD</td>
</tr>
<tr>
<td>32</td>
<td>SAM</td>
</tr>
<tr>
<td>33</td>
<td>APM</td>
</tr>
<tr>
<td>34</td>
<td>MRM</td>
</tr>
<tr>
<td>35</td>
<td>TRANID</td>
</tr>
<tr>
<td>36</td>
<td>TERMID</td>
</tr>
<tr>
<td>37</td>
<td>SIGNID</td>
</tr>
<tr>
<td>38</td>
<td>RACFGID</td>
</tr>
<tr>
<td>39</td>
<td>USERID</td>
</tr>
<tr>
<td>40</td>
<td>NULL</td>
</tr>
<tr>
<td>41</td>
<td>CHARSTR</td>
</tr>
<tr>
<td>42</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>43</td>
<td>INACTIVE</td>
</tr>
<tr>
<td>44</td>
<td>WAITING</td>
</tr>
<tr>
<td>45</td>
<td>QUIESCING</td>
</tr>
<tr>
<td>46</td>
<td>POOL</td>
</tr>
<tr>
<td>47</td>
<td>LTRAN</td>
</tr>
<tr>
<td>48</td>
<td>RTRAN</td>
</tr>
<tr>
<td>49</td>
<td>GOAL</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>54</td>
<td>QUEUE</td>
</tr>
<tr>
<td>55</td>
<td>LUNAME</td>
</tr>
<tr>
<td>58</td>
<td>DELIMIT</td>
</tr>
<tr>
<td>59</td>
<td>PCONV</td>
</tr>
<tr>
<td>60</td>
<td>LOGON</td>
</tr>
<tr>
<td>61</td>
<td>SIGNON</td>
</tr>
<tr>
<td>63</td>
<td>PERMANENT</td>
</tr>
<tr>
<td>64</td>
<td>MCICS</td>
</tr>
<tr>
<td>65</td>
<td>MGLBL</td>
</tr>
<tr>
<td>66</td>
<td>MDBX</td>
</tr>
<tr>
<td>67</td>
<td>MCONN</td>
</tr>
<tr>
<td>68</td>
<td>MFILE</td>
</tr>
<tr>
<td>69</td>
<td>MJRNCL</td>
</tr>
<tr>
<td>70</td>
<td>MPROG</td>
</tr>
<tr>
<td>71</td>
<td>MTERM</td>
</tr>
<tr>
<td>72</td>
<td>MTDQS</td>
</tr>
<tr>
<td>73</td>
<td>MTRAN</td>
</tr>
<tr>
<td>74</td>
<td>MAPPL</td>
</tr>
<tr>
<td>75</td>
<td>ABOVE</td>
</tr>
<tr>
<td>76</td>
<td>BELOW</td>
</tr>
<tr>
<td>77</td>
<td>NOCOPY</td>
</tr>
<tr>
<td>78</td>
<td>DSA</td>
</tr>
<tr>
<td>79</td>
<td>CDSA</td>
</tr>
<tr>
<td>80</td>
<td>UDSA</td>
</tr>
<tr>
<td>81</td>
<td>LPA</td>
</tr>
<tr>
<td>82</td>
<td>EDSA</td>
</tr>
<tr>
<td>83</td>
<td>ECDSA</td>
</tr>
<tr>
<td>84</td>
<td>EUDSA</td>
</tr>
<tr>
<td>85</td>
<td>ERDSA</td>
</tr>
<tr>
<td>86</td>
<td>ELPA</td>
</tr>
<tr>
<td>87</td>
<td>CICS</td>
</tr>
<tr>
<td>88</td>
<td>USER</td>
</tr>
<tr>
<td>89</td>
<td>READONLY</td>
</tr>
<tr>
<td>90</td>
<td>LU61</td>
</tr>
<tr>
<td>91</td>
<td>LU62</td>
</tr>
<tr>
<td>92</td>
<td>INDIRECT</td>
</tr>
<tr>
<td>93</td>
<td>MRO</td>
</tr>
</tbody>
</table>

Numeric EYUDA general values
### Numeric EYUDA general values

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>94</td>
<td>NOTAPPLIC</td>
</tr>
<tr>
<td>95</td>
<td>LFILE</td>
</tr>
<tr>
<td>96</td>
<td>RFILE</td>
</tr>
<tr>
<td>97</td>
<td>CTABL</td>
</tr>
<tr>
<td>98</td>
<td>UTABL</td>
</tr>
<tr>
<td>99</td>
<td>INSTALLED</td>
</tr>
<tr>
<td>100</td>
<td>PENDING</td>
</tr>
<tr>
<td>101</td>
<td>INHERIT</td>
</tr>
<tr>
<td>102</td>
<td>EXPLICIT</td>
</tr>
<tr>
<td>103</td>
<td>CICSSYS</td>
</tr>
<tr>
<td>104</td>
<td>SYSGROUP</td>
</tr>
<tr>
<td>105</td>
<td>KEEP</td>
</tr>
<tr>
<td>106</td>
<td>NAME</td>
</tr>
<tr>
<td>107</td>
<td>FORCE</td>
</tr>
<tr>
<td>108</td>
<td>NONE</td>
</tr>
<tr>
<td>109</td>
<td>UNASSIGNED</td>
</tr>
<tr>
<td>110</td>
<td>DROP</td>
</tr>
<tr>
<td>111</td>
<td>LOCAL</td>
</tr>
<tr>
<td>112</td>
<td>REMOTE</td>
</tr>
<tr>
<td>113</td>
<td>DEFAULT</td>
</tr>
<tr>
<td>114</td>
<td>REMOVE</td>
</tr>
<tr>
<td>115</td>
<td>DORMANT</td>
</tr>
<tr>
<td>116</td>
<td>START</td>
</tr>
<tr>
<td>117</td>
<td>END</td>
</tr>
<tr>
<td>118</td>
<td>ADJACENT</td>
</tr>
<tr>
<td>119</td>
<td>LOSTCON</td>
</tr>
<tr>
<td>120</td>
<td>CREATING</td>
</tr>
<tr>
<td>121</td>
<td>REMOVING</td>
</tr>
<tr>
<td>122</td>
<td>QUIESCED</td>
</tr>
<tr>
<td>123</td>
<td>LINKACTIVE</td>
</tr>
<tr>
<td>124</td>
<td>LINKDOWN</td>
</tr>
<tr>
<td>125</td>
<td>ESSS</td>
</tr>
<tr>
<td>126</td>
<td>CONACT</td>
</tr>
<tr>
<td>127</td>
<td>RESET</td>
</tr>
<tr>
<td>128</td>
<td>SYSDUMP</td>
</tr>
<tr>
<td>129</td>
<td>TRANDUMP</td>
</tr>
<tr>
<td>130</td>
<td>MAXTASK</td>
</tr>
<tr>
<td></td>
<td>Numeric EYUDA general values</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>131</td>
<td>STALLED</td>
</tr>
<tr>
<td>132</td>
<td>SOSUDSA</td>
</tr>
<tr>
<td>133</td>
<td>SOSCDSA</td>
</tr>
<tr>
<td>134</td>
<td>SOSEUDSA</td>
</tr>
<tr>
<td>135</td>
<td>SOSECDSA</td>
</tr>
<tr>
<td>136</td>
<td>SOSERDSA</td>
</tr>
<tr>
<td>137</td>
<td>SOSSDSA</td>
</tr>
<tr>
<td>138</td>
<td>SOSESDSA</td>
</tr>
<tr>
<td>139</td>
<td>SOSRDSA</td>
</tr>
<tr>
<td>140</td>
<td>QUIESCE</td>
</tr>
<tr>
<td>141</td>
<td>PRIMARY</td>
</tr>
<tr>
<td>142</td>
<td>SECONDARY</td>
</tr>
<tr>
<td>143</td>
<td>DUPLICATE</td>
</tr>
<tr>
<td>144</td>
<td>FROZEN</td>
</tr>
<tr>
<td>145</td>
<td>ALL</td>
</tr>
<tr>
<td>146</td>
<td>ANY</td>
</tr>
<tr>
<td>147</td>
<td>SUM</td>
</tr>
<tr>
<td>148</td>
<td>MIN</td>
</tr>
<tr>
<td>149</td>
<td>MAX</td>
</tr>
<tr>
<td>150</td>
<td>AVG</td>
</tr>
<tr>
<td>151</td>
<td>CNT</td>
</tr>
<tr>
<td>152</td>
<td>LE</td>
</tr>
<tr>
<td>153</td>
<td>GE</td>
</tr>
<tr>
<td>154</td>
<td>SDSA</td>
</tr>
<tr>
<td>155</td>
<td>ESDSA</td>
</tr>
<tr>
<td>156</td>
<td>RDSA</td>
</tr>
<tr>
<td>161</td>
<td>EXECUTE</td>
</tr>
<tr>
<td>162</td>
<td>CHECK</td>
</tr>
<tr>
<td>163</td>
<td>LOSTCMAS</td>
</tr>
<tr>
<td>164</td>
<td>LOSTMAS</td>
</tr>
<tr>
<td>165</td>
<td>AASTERISK</td>
</tr>
<tr>
<td>166</td>
<td>BLANK</td>
</tr>
<tr>
<td>167</td>
<td>INDEX</td>
</tr>
<tr>
<td>168</td>
<td>DATA</td>
</tr>
<tr>
<td>169</td>
<td>BOTH</td>
</tr>
<tr>
<td>170</td>
<td>NETBIOS</td>
</tr>
<tr>
<td>171</td>
<td>TCPIP</td>
</tr>
</tbody>
</table>
Numeric EYUDA general values

172  AFTER
173  ALLREQS
174  ASA
177  ASIS
178  ASSEMBLER
179  BACKOUTONLY
180  BEFORE
181  BLUE
182  C
183  CLEARCONV
184  CLOSE
185  COBOL
186  COLD
187  CYCLIC
188  DEFERRED
189  DIP
190  DISK
191  EODS
192  EXTA
193  FILE
194  FIRSTREF
195  GREEN
196  IDENTIFY
197  IGNORE
198  INITIAL
199  INOUT
200  INPUT
201  LEAVE
202  LE370
203  LINEAR
204  LINK
205  LMS
206  LOGICAL
207  LOGOFF
208  LRU
209  MACHINE
210  MESSAGE
Numeric EYUDA general values

211 MIXIDPE
212 MOD
213 MODIFYREQS
214 MSRE
215 NEUTRAL
216 NEW
217 NOFORCE
218 NONVTAM
219 OLD
220 ONLY
221 OPEN
222 OPI
223 OUTPUT
224 PERSISTENT
225 PHYSICAL
226 PINK
227 PLI
228 PRINTER
229 RED
230 RECOVERY
231 REJECT
232 RELEASESESS
233 REREAD
234 RPG
235 SCS
236 SECURITY
237 SHR
238 SKIP
239 SPECIFIC
240 STARTIO
241 STARTUP
242 STRFIELD
243 SYSDEFAULT
244 TAPE
245 TERMINAL
246 TRANSACTION
247 TRANSIENT
Numeric EYUDA general values

248  TURQUOISE
249  U
250  UNCONDREL
251  UPDATEONLY
252  VB
253  VERIFY
254  VTAM
255  YELLOW
256  3270
257  AUTO
258  DYNAM
259  EXTRA
260  INTRA
261  IND
262  STAT
263  RELATED
264  TARGET
265  NEVER
266  ALWAYS
267  COLDONLY
268  WARMONLY
269  PROMPT
270  CONTINUE
271  TERMINATE
272  SHUTDOWN
273  RTADEF
274  STATDEF
275  CONNDEF
276  FILEDEF
277  JRNLDEF
278  JRNMDEF
279  LSRDEF
280  MAPDEF
281  PARTDEF
282  PRDNDEF
283  PROFDEF
284  PROGDEF
Numeric EYUDA general values

285  SESSDEF
286  TDQDEF
287  TERMDEF
288  TRANDEF
289  TRNCLDEF
290  TSQDEF
291  TYPTMDEF
292  MPSYNCCR
293  ASSOCIATIONS
294  MEMBERS
295  DB2CDEF
296  DB2EDEF
297  DB2TDEF
298  FSEGDEF
299  TSMDEF
300  ENQMDEF
301  TCPDEF
302  DOCDEF
303  FULL
304  RELEASE
305  PA1
306  PA2
307  PA3
308  PF1
309  PF2
310  PF3
311  PF4
312  PF5
313  PF6
314  PF7
315  PF8
316  PF9
317  PF10
318  PF11
319  PF12
320  PF13
321  PF14
### Numeric EYUDA general values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>322</td>
<td>PF15</td>
</tr>
<tr>
<td>323</td>
<td>PF16</td>
</tr>
<tr>
<td>324</td>
<td>PF17</td>
</tr>
<tr>
<td>325</td>
<td>PF18</td>
</tr>
<tr>
<td>326</td>
<td>PF19</td>
</tr>
<tr>
<td>327</td>
<td>PF20</td>
</tr>
<tr>
<td>328</td>
<td>PF21</td>
</tr>
<tr>
<td>329</td>
<td>PF22</td>
</tr>
<tr>
<td>330</td>
<td>PF23</td>
</tr>
<tr>
<td>331</td>
<td>PF24</td>
</tr>
<tr>
<td>332</td>
<td>STANDARD</td>
</tr>
<tr>
<td>333</td>
<td>APPC</td>
</tr>
<tr>
<td>334</td>
<td>BATCHDI</td>
</tr>
<tr>
<td>335</td>
<td>BCHLU</td>
</tr>
<tr>
<td>336</td>
<td>CONSOLE</td>
</tr>
<tr>
<td>337</td>
<td>CONTLU</td>
</tr>
<tr>
<td>338</td>
<td>INTLU</td>
</tr>
<tr>
<td>339</td>
<td>LUTYPE2</td>
</tr>
<tr>
<td>340</td>
<td>LUTYPE3</td>
</tr>
<tr>
<td>341</td>
<td>LUTYPE4</td>
</tr>
<tr>
<td>342</td>
<td>L3277</td>
</tr>
<tr>
<td>343</td>
<td>L3284</td>
</tr>
<tr>
<td>344</td>
<td>L3286</td>
</tr>
<tr>
<td>345</td>
<td>PIPELINE</td>
</tr>
<tr>
<td>346</td>
<td>SCSPRINT</td>
</tr>
<tr>
<td>347</td>
<td>TLX</td>
</tr>
<tr>
<td>348</td>
<td>TWX</td>
</tr>
<tr>
<td>349</td>
<td>USERPROG</td>
</tr>
<tr>
<td>350</td>
<td>3270P</td>
</tr>
<tr>
<td>351</td>
<td>3275</td>
</tr>
<tr>
<td>352</td>
<td>3277</td>
</tr>
<tr>
<td>353</td>
<td>3277CM</td>
</tr>
<tr>
<td>354</td>
<td>3284</td>
</tr>
<tr>
<td>355</td>
<td>3284CM</td>
</tr>
<tr>
<td>356</td>
<td>3286</td>
</tr>
<tr>
<td>357</td>
<td>3286CM</td>
</tr>
<tr>
<td>358</td>
<td>3600</td>
</tr>
<tr>
<td>359</td>
<td>3614</td>
</tr>
<tr>
<td>360</td>
<td>3650</td>
</tr>
<tr>
<td>361</td>
<td>3653</td>
</tr>
<tr>
<td>362</td>
<td>3767</td>
</tr>
<tr>
<td>363</td>
<td>3767C</td>
</tr>
<tr>
<td>364</td>
<td>3767I</td>
</tr>
<tr>
<td>365</td>
<td>3770</td>
</tr>
<tr>
<td>366</td>
<td>3770B</td>
</tr>
<tr>
<td>367</td>
<td>3770C</td>
</tr>
<tr>
<td>368</td>
<td>3770I</td>
</tr>
<tr>
<td>369</td>
<td>3790</td>
</tr>
<tr>
<td>370</td>
<td>BLINK</td>
</tr>
<tr>
<td>371</td>
<td>REVERSE</td>
</tr>
<tr>
<td>372</td>
<td>UNDERLINE</td>
</tr>
<tr>
<td>373</td>
<td>INSTALL</td>
</tr>
<tr>
<td>374</td>
<td>RESDEF</td>
</tr>
<tr>
<td>375</td>
<td>RASINDSC</td>
</tr>
<tr>
<td>376</td>
<td>RESTYPE</td>
</tr>
<tr>
<td>377</td>
<td>SCOPETYP</td>
</tr>
<tr>
<td>378</td>
<td>REBUILD</td>
</tr>
<tr>
<td>379</td>
<td>RECONNECT</td>
</tr>
<tr>
<td>380</td>
<td>CONNECTING</td>
</tr>
<tr>
<td>381</td>
<td>TWAIT</td>
</tr>
<tr>
<td>382</td>
<td>NOTWAIT</td>
</tr>
<tr>
<td>383</td>
<td>DISCONNING</td>
</tr>
<tr>
<td>384</td>
<td>CFTBL</td>
</tr>
<tr>
<td>385</td>
<td>CF</td>
</tr>
<tr>
<td>386</td>
<td>QUASIRENT</td>
</tr>
<tr>
<td>387</td>
<td>THREADSAFE</td>
</tr>
<tr>
<td>388</td>
<td>PROCDEF</td>
</tr>
<tr>
<td>389</td>
<td>BAPPL</td>
</tr>
<tr>
<td>390</td>
<td>ACTIVITY</td>
</tr>
<tr>
<td>391</td>
<td>PROCESS</td>
</tr>
<tr>
<td>393</td>
<td>PMWINDOW</td>
</tr>
<tr>
<td>394</td>
<td>FULLSCREEN</td>
</tr>
<tr>
<td>395</td>
<td>COM1</td>
</tr>
<tr>
<td>396</td>
<td>COM2</td>
</tr>
</tbody>
</table>
Numeric EYUDA general values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>397</td>
<td>COM3</td>
</tr>
<tr>
<td>398</td>
<td>COM4</td>
</tr>
<tr>
<td>399</td>
<td>COM5</td>
</tr>
<tr>
<td>400</td>
<td>COM6</td>
</tr>
<tr>
<td>401</td>
<td>COM7</td>
</tr>
<tr>
<td>402</td>
<td>COM8</td>
</tr>
<tr>
<td>403</td>
<td>CLIENTAUTH</td>
</tr>
<tr>
<td>404</td>
<td>RQMDEF</td>
</tr>
<tr>
<td>405</td>
<td>3270TERM</td>
</tr>
<tr>
<td>406</td>
<td>3270PRNT</td>
</tr>
<tr>
<td>407</td>
<td>3270DBTM</td>
</tr>
<tr>
<td>408</td>
<td>3270DBPR</td>
</tr>
<tr>
<td>409</td>
<td>3151TERM</td>
</tr>
<tr>
<td>410</td>
<td>SEQTERM</td>
</tr>
<tr>
<td>411</td>
<td>EOF</td>
</tr>
<tr>
<td>412</td>
<td>EOT</td>
</tr>
<tr>
<td>413</td>
<td>EPOODEF</td>
</tr>
<tr>
<td>414</td>
<td>FETRGDEF</td>
</tr>
<tr>
<td>415</td>
<td>FENODDEF</td>
</tr>
<tr>
<td>416</td>
<td>FEPRODEF</td>
</tr>
<tr>
<td>417</td>
<td>DEBUG</td>
</tr>
<tr>
<td>418</td>
<td>LEVSE</td>
</tr>
<tr>
<td>419</td>
<td>KEY</td>
</tr>
<tr>
<td>420</td>
<td>RBA</td>
</tr>
<tr>
<td>421</td>
<td>EJCODEF</td>
</tr>
<tr>
<td>422</td>
<td>EJDJDEF</td>
</tr>
<tr>
<td>423</td>
<td>BASIC</td>
</tr>
<tr>
<td>424</td>
<td>CERTIFICATE</td>
</tr>
<tr>
<td>425</td>
<td>CLIENTCERT</td>
</tr>
</tbody>
</table>

EYUDA general values in alphabetic order

This section lists the general EYUDAs in alphabetic order by their character values.

<table>
<thead>
<tr>
<th>EYUDA</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AASTERISK</td>
<td>165</td>
</tr>
<tr>
<td>Term</td>
<td>Page</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
</tr>
<tr>
<td>ABOVE</td>
<td>75</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>45</td>
</tr>
<tr>
<td>ACTIVITY</td>
<td>390</td>
</tr>
<tr>
<td>ADJACENT</td>
<td>118</td>
</tr>
<tr>
<td>AFTER</td>
<td>172</td>
</tr>
<tr>
<td>ALL</td>
<td>145</td>
</tr>
<tr>
<td>ALLREQS</td>
<td>173</td>
</tr>
<tr>
<td>ALWAYS</td>
<td>266</td>
</tr>
<tr>
<td>ANY</td>
<td>146</td>
</tr>
<tr>
<td>APM</td>
<td>36</td>
</tr>
<tr>
<td>APPC</td>
<td>333</td>
</tr>
<tr>
<td>ASA</td>
<td>174</td>
</tr>
<tr>
<td>ASIS</td>
<td>177</td>
</tr>
<tr>
<td>ASSEMBLER</td>
<td>178</td>
</tr>
<tr>
<td>ASSOCIATIONS</td>
<td>293</td>
</tr>
<tr>
<td>AUTO</td>
<td>257</td>
</tr>
<tr>
<td>AUTOMATIC</td>
<td>428</td>
</tr>
<tr>
<td>AUTOREGISTER</td>
<td>427</td>
</tr>
<tr>
<td>AVG</td>
<td>150</td>
</tr>
<tr>
<td>BACKOUTONLY</td>
<td>179</td>
</tr>
<tr>
<td>BAPPL</td>
<td>389</td>
</tr>
<tr>
<td>BASIC</td>
<td>425</td>
</tr>
<tr>
<td>BATCHDI</td>
<td>334</td>
</tr>
<tr>
<td>BCHLU</td>
<td>335</td>
</tr>
<tr>
<td>BEFORE</td>
<td>180</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

BELOW
  76
BLANK
  166
BLINK
  370
BLUE
  181
BOTH
  169
C
  182
CDSA
  79
CERTIFICATE
  426
CF
  385
CFTBL
  384
CHARSTR
  44
CHECK
  162
CICS
  87
CICSSYS
  103
CLEARCONV
  183
CLIENTAUTH
  403
CLIENTCERT
  429
CLOSE
  184
CNT
  151
COBOL
  185
COLD
  186
COLDONLY
  267
COM1
  395
COM2
  396
COM3
  397
COM4
  398
COM5
  399
COM6
  400
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>COM7</td>
<td>401</td>
</tr>
<tr>
<td>COM8</td>
<td>402</td>
</tr>
<tr>
<td>CONACT</td>
<td>126</td>
</tr>
<tr>
<td>CONNDEF</td>
<td>275</td>
</tr>
<tr>
<td>CONNECTING</td>
<td>380</td>
</tr>
<tr>
<td>CONSOLE</td>
<td>336</td>
</tr>
<tr>
<td>CONTINUE</td>
<td>270</td>
</tr>
<tr>
<td>CONLU</td>
<td>337</td>
</tr>
<tr>
<td>CREATING</td>
<td>120</td>
</tr>
<tr>
<td>CTABLE</td>
<td>97</td>
</tr>
<tr>
<td>CYCLIC</td>
<td>187</td>
</tr>
<tr>
<td>DATA</td>
<td>168</td>
</tr>
<tr>
<td>DB2CDEF</td>
<td>295</td>
</tr>
<tr>
<td>DB2EDEF</td>
<td>296</td>
</tr>
<tr>
<td>DB2TDEF</td>
<td>297</td>
</tr>
<tr>
<td>DEBUG</td>
<td>417</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>113</td>
</tr>
<tr>
<td>DEFERRED</td>
<td>188</td>
</tr>
<tr>
<td>DELIMIT</td>
<td>58</td>
</tr>
<tr>
<td>DIP</td>
<td>189</td>
</tr>
<tr>
<td>DISCONNING</td>
<td>383</td>
</tr>
<tr>
<td>DISK</td>
<td>190</td>
</tr>
<tr>
<td>DOCDEF</td>
<td>302</td>
</tr>
<tr>
<td>DORMANT</td>
<td>115</td>
</tr>
<tr>
<td>DROP</td>
<td>110</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

DSA 78
DUPLICATE 143
DYNAM 258
ECDSA 83
EDSA 82
EJCODEF 423
EJDJDEF 424
ELPA 86
END 117
ENQMDEF 300
EODS 191
EOF 411
EOT 412
EQ 16
ERDSA 85
ESDSA 155
ESSS 125
EUDSA 84
EXECUTE 161
EXPLICIT 102
EXTA 192
EXTRA 259
FALSE 8
FENODDEF 415
FEPOODEF 413
FEPRODEF 416
<table>
<thead>
<tr>
<th>EYUDA General Values</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FETRGDEF</td>
<td>414</td>
</tr>
<tr>
<td>FILE</td>
<td>193</td>
</tr>
<tr>
<td>FILEDEF</td>
<td>276</td>
</tr>
<tr>
<td>FIRSTREF</td>
<td>194</td>
</tr>
<tr>
<td>FORCE</td>
<td>107</td>
</tr>
<tr>
<td>FROZEN</td>
<td>144</td>
</tr>
<tr>
<td>FSEGDEF</td>
<td>298</td>
</tr>
<tr>
<td>FULL</td>
<td>303</td>
</tr>
<tr>
<td>FULLSCREEN</td>
<td>394</td>
</tr>
<tr>
<td>GE</td>
<td>153</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>27</td>
</tr>
<tr>
<td>GOAL</td>
<td>53</td>
</tr>
<tr>
<td>GREEN</td>
<td>195</td>
</tr>
<tr>
<td>GT</td>
<td>18</td>
</tr>
<tr>
<td>HIGH</td>
<td>21</td>
</tr>
<tr>
<td>HS</td>
<td>14</td>
</tr>
<tr>
<td>HW</td>
<td>13</td>
</tr>
<tr>
<td>IDENTIFY</td>
<td>196</td>
</tr>
<tr>
<td>IGNORE</td>
<td>197</td>
</tr>
<tr>
<td>IMMEDIATE</td>
<td>23</td>
</tr>
<tr>
<td>INACTIVE</td>
<td>46</td>
</tr>
<tr>
<td>IND</td>
<td>261</td>
</tr>
<tr>
<td>INDEX</td>
<td>167</td>
</tr>
<tr>
<td>INDIRECT</td>
<td>92</td>
</tr>
<tr>
<td>INHERIT</td>
<td>101</td>
</tr>
<tr>
<td>INITIAL</td>
<td>198</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INOUT</td>
<td>199</td>
</tr>
<tr>
<td>INPUT</td>
<td>200</td>
</tr>
<tr>
<td>INSTALL</td>
<td>373</td>
</tr>
<tr>
<td>INSTALLED</td>
<td>99</td>
</tr>
<tr>
<td>INTLU</td>
<td>338</td>
</tr>
<tr>
<td>INTRA</td>
<td>260</td>
</tr>
<tr>
<td>INVALID</td>
<td>6</td>
</tr>
<tr>
<td>JRNLDEF</td>
<td>277</td>
</tr>
<tr>
<td>JRMNDEF</td>
<td>278</td>
</tr>
<tr>
<td>KEEP</td>
<td>105</td>
</tr>
<tr>
<td>KEY</td>
<td>419</td>
</tr>
<tr>
<td>LE</td>
<td>152</td>
</tr>
<tr>
<td>LEAVE</td>
<td>201</td>
</tr>
<tr>
<td>LEVSE</td>
<td>418</td>
</tr>
<tr>
<td>LE370</td>
<td>202</td>
</tr>
<tr>
<td>LFILE</td>
<td>95</td>
</tr>
<tr>
<td>LINEAR</td>
<td>203</td>
</tr>
<tr>
<td>LINK</td>
<td>204</td>
</tr>
<tr>
<td>LINKACTIVE</td>
<td>123</td>
</tr>
<tr>
<td>LINKDOWN</td>
<td>124</td>
</tr>
<tr>
<td>LMS</td>
<td>205</td>
</tr>
<tr>
<td>LOCAL</td>
<td>111</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>206</td>
</tr>
<tr>
<td>LOGOFF</td>
<td>207</td>
</tr>
<tr>
<td>LOGON</td>
<td>207</td>
</tr>
<tr>
<td>LOSTCMAS</td>
<td>163</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

LOSTCON 119
LOSTMAS 164
LOW 20
LPA 81
LRU 208
LS 10
LSRDEF 279
LT 19
LTRAN 50
LUNAME 55
LUTYPE2 339
LUTYPE3 340
LUTYPE4 341
LU61 90
LU62 91
LW 11
L3277 342
L3284 343
L3286 344
MACHINE 209
MAPDEF 280
MAPPL 74
MAX 149
MAXTASK 130
MCICS 64
MCONN 67
MDBX 66
### Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMBERS</td>
<td>294</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>210</td>
</tr>
<tr>
<td>MFILE</td>
<td>68</td>
</tr>
<tr>
<td>MGLBL</td>
<td>65</td>
</tr>
<tr>
<td>MIN</td>
<td>148</td>
</tr>
<tr>
<td>MIXIDPE</td>
<td>211</td>
</tr>
<tr>
<td>MJRNL</td>
<td>69</td>
</tr>
<tr>
<td>MOD</td>
<td>212</td>
</tr>
<tr>
<td>MODIFYREQS</td>
<td>213</td>
</tr>
<tr>
<td>MPROG</td>
<td>70</td>
</tr>
<tr>
<td>MPSYNCCR</td>
<td>292</td>
</tr>
<tr>
<td>MRM</td>
<td>37</td>
</tr>
<tr>
<td>MRO</td>
<td>93</td>
</tr>
<tr>
<td>MSRE</td>
<td>214</td>
</tr>
<tr>
<td>MTDQS</td>
<td>72</td>
</tr>
<tr>
<td>MTERM</td>
<td>71</td>
</tr>
<tr>
<td>MTRAN</td>
<td>73</td>
</tr>
<tr>
<td>N_A</td>
<td>0</td>
</tr>
<tr>
<td>NA</td>
<td>0</td>
</tr>
<tr>
<td>NAME</td>
<td>106</td>
</tr>
<tr>
<td>NE</td>
<td>17</td>
</tr>
<tr>
<td>NETBIOS</td>
<td>170</td>
</tr>
<tr>
<td>NEUTRAL</td>
<td>215</td>
</tr>
<tr>
<td>NEVER</td>
<td>265</td>
</tr>
<tr>
<td>NEW</td>
<td>216</td>
</tr>
<tr>
<td>NM</td>
<td>12</td>
</tr>
<tr>
<td>NO</td>
<td>2</td>
</tr>
<tr>
<td>Value</td>
<td>Page</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>NOCOPY</td>
<td>77</td>
</tr>
<tr>
<td>NOFORCE</td>
<td>217</td>
</tr>
<tr>
<td>NONE</td>
<td>108</td>
</tr>
<tr>
<td>NONVTAM</td>
<td>218</td>
</tr>
<tr>
<td>NORMAL</td>
<td>22</td>
</tr>
<tr>
<td>NOSHUT</td>
<td>26</td>
</tr>
<tr>
<td>NOTAPPLIC</td>
<td>94</td>
</tr>
<tr>
<td>NOTWAIT</td>
<td>382</td>
</tr>
<tr>
<td>NULL</td>
<td>43</td>
</tr>
<tr>
<td>OFF</td>
<td>4</td>
</tr>
<tr>
<td>OLD</td>
<td>219</td>
</tr>
<tr>
<td>ON</td>
<td>3</td>
</tr>
<tr>
<td>ONLY</td>
<td>220</td>
</tr>
<tr>
<td>OPEN</td>
<td>221</td>
</tr>
<tr>
<td>OPID</td>
<td>222</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>223</td>
</tr>
<tr>
<td>PARTDEF</td>
<td>281</td>
</tr>
<tr>
<td>PA1</td>
<td>305</td>
</tr>
<tr>
<td>PA2</td>
<td>306</td>
</tr>
<tr>
<td>PA3</td>
<td>307</td>
</tr>
<tr>
<td>PCONV</td>
<td>59</td>
</tr>
<tr>
<td>PENDING</td>
<td>100</td>
</tr>
<tr>
<td>PERMANENT</td>
<td>63</td>
</tr>
<tr>
<td>PERSISTENT</td>
<td>224</td>
</tr>
<tr>
<td>PF1</td>
<td>308</td>
</tr>
<tr>
<td>PF10</td>
<td>317</td>
</tr>
<tr>
<td>PF11</td>
<td>318</td>
</tr>
<tr>
<td>PF12</td>
<td>319</td>
</tr>
<tr>
<td>PF13</td>
<td>320</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF14</td>
<td>321</td>
</tr>
<tr>
<td>PF15</td>
<td>322</td>
</tr>
<tr>
<td>PF16</td>
<td>323</td>
</tr>
<tr>
<td>PF17</td>
<td>324</td>
</tr>
<tr>
<td>PF18</td>
<td>325</td>
</tr>
<tr>
<td>PF19</td>
<td>326</td>
</tr>
<tr>
<td>PF2</td>
<td>309</td>
</tr>
<tr>
<td>PF20</td>
<td>327</td>
</tr>
<tr>
<td>PF21</td>
<td>328</td>
</tr>
<tr>
<td>PF22</td>
<td>329</td>
</tr>
<tr>
<td>PF23</td>
<td>330</td>
</tr>
<tr>
<td>PF24</td>
<td>331</td>
</tr>
<tr>
<td>PF3</td>
<td>310</td>
</tr>
<tr>
<td>PF4</td>
<td>311</td>
</tr>
<tr>
<td>PF5</td>
<td>312</td>
</tr>
<tr>
<td>PF6</td>
<td>313</td>
</tr>
<tr>
<td>PF7</td>
<td>314</td>
</tr>
<tr>
<td>PF8</td>
<td>315</td>
</tr>
<tr>
<td>PF9</td>
<td>316</td>
</tr>
<tr>
<td>PHYSICAL</td>
<td>225</td>
</tr>
<tr>
<td>PINK</td>
<td>226</td>
</tr>
<tr>
<td>PIPELINE</td>
<td>345</td>
</tr>
<tr>
<td>PLI</td>
<td>227</td>
</tr>
<tr>
<td>PMWINDOW</td>
<td>393</td>
</tr>
<tr>
<td>POOL</td>
<td>49</td>
</tr>
<tr>
<td>PRIMARY</td>
<td>141</td>
</tr>
<tr>
<td>PRINTER</td>
<td>228</td>
</tr>
<tr>
<td>PROCDEF</td>
<td>388</td>
</tr>
<tr>
<td>PROCESS</td>
<td>391</td>
</tr>
<tr>
<td>PROFDEF</td>
<td>283</td>
</tr>
<tr>
<td>PROGDEF</td>
<td>284</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

PROMPT 269
PRTNDEF 282
QUASIRENT 386
QUEUE 54
QUIESCE 140
QUIESCED 122
QUIESCING 48
RACFGID 41
RASINDSC 375
RBA 420
RDSA 156
READONLY 89
REBUILD 378
RECONNECT 379
RECOVERY 230
RED 229
REJECT 231
RELATED 263
RELEASE 304
RELEASESESS 232
REMOTE 112
REMOVE 114
REMOVING 121
Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>REREAD</td>
<td>233</td>
</tr>
<tr>
<td>RESDEF</td>
<td>374</td>
</tr>
<tr>
<td>RESET</td>
<td>127</td>
</tr>
<tr>
<td>RESTYPE</td>
<td>376</td>
</tr>
<tr>
<td>REVERSE</td>
<td>371</td>
</tr>
<tr>
<td>RFILE</td>
<td>96</td>
</tr>
<tr>
<td>RPG</td>
<td>234</td>
</tr>
<tr>
<td>RQMDEF</td>
<td>404</td>
</tr>
<tr>
<td>RTADEF</td>
<td>273</td>
</tr>
<tr>
<td>RTRAN</td>
<td>51</td>
</tr>
<tr>
<td>SAM</td>
<td>35</td>
</tr>
<tr>
<td>SCOPETYP</td>
<td>377</td>
</tr>
<tr>
<td>SCS</td>
<td>235</td>
</tr>
<tr>
<td>SCSPRINT</td>
<td>346</td>
</tr>
<tr>
<td>SDSA</td>
<td>154</td>
</tr>
<tr>
<td>SECONDARY</td>
<td>142</td>
</tr>
<tr>
<td>SECURITY</td>
<td>236</td>
</tr>
<tr>
<td>SEQTERM</td>
<td>410</td>
</tr>
<tr>
<td>SESSDEF</td>
<td>285</td>
</tr>
<tr>
<td>SHR</td>
<td>237</td>
</tr>
<tr>
<td>SHUT</td>
<td>25</td>
</tr>
<tr>
<td>SHUTDOWN</td>
<td>272</td>
</tr>
<tr>
<td>SIGNID</td>
<td>40</td>
</tr>
<tr>
<td>SIGNON</td>
<td>61</td>
</tr>
<tr>
<td>SKIP</td>
<td>238</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

SOSCDSA 133
SOSECDSA 135
SOSERDSA 136
SOSESDSA 138
SOSEUDSA 134
SOSRDSA 139
SOSSDSA 137
SOSUDSA 132
SPECIFIC 239
STALLED 131
STANDARD 332
START 116
STARTIO 240
STARTUP 241
STAT 262
STATDEF 274
STRFIELD 242
SUM 147
SUSPEND 30
SYSDEFAULT 243
SYSDUMP 128
SYSGROUP 104
SYSTEM 28
Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAKEOVER</td>
<td>24</td>
</tr>
<tr>
<td>TAPE</td>
<td>244</td>
</tr>
<tr>
<td>TARGET</td>
<td>264</td>
</tr>
<tr>
<td>TCPDEF</td>
<td>301</td>
</tr>
<tr>
<td>TCPIP</td>
<td>171</td>
</tr>
<tr>
<td>TDQDEF</td>
<td>286</td>
</tr>
<tr>
<td>TERMDEF</td>
<td>287</td>
</tr>
<tr>
<td>TERMID</td>
<td>39</td>
</tr>
<tr>
<td>TERMINAL</td>
<td>245</td>
</tr>
<tr>
<td>TERMINATE</td>
<td>271</td>
</tr>
<tr>
<td>THREADSAFE</td>
<td>387</td>
</tr>
<tr>
<td>THRESHOLD</td>
<td>34</td>
</tr>
<tr>
<td>TLX</td>
<td>347</td>
</tr>
<tr>
<td>TRANDEF</td>
<td>288</td>
</tr>
<tr>
<td>TRANDUMP</td>
<td>129</td>
</tr>
<tr>
<td>TRANID</td>
<td>38</td>
</tr>
<tr>
<td>TRANSACTION</td>
<td>246</td>
</tr>
<tr>
<td>TRANSIENT</td>
<td>247</td>
</tr>
<tr>
<td>TRNCLDEF</td>
<td>289</td>
</tr>
<tr>
<td>TRUE</td>
<td>7</td>
</tr>
<tr>
<td>TSMDEF</td>
<td>299</td>
</tr>
<tr>
<td>TSQDEF</td>
<td>290</td>
</tr>
<tr>
<td>TURQUOISE</td>
<td>248</td>
</tr>
<tr>
<td>TWAIT</td>
<td>381</td>
</tr>
<tr>
<td>TWX</td>
<td>348</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>TYPTMDEF</td>
<td>291</td>
</tr>
<tr>
<td>U</td>
<td>249</td>
</tr>
<tr>
<td>USDA</td>
<td>80</td>
</tr>
<tr>
<td>UNASSIGNED</td>
<td>109</td>
</tr>
<tr>
<td>UNCONDREL</td>
<td>250</td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>372</td>
</tr>
<tr>
<td>UPDATEONLY</td>
<td>251</td>
</tr>
<tr>
<td>USER</td>
<td>88</td>
</tr>
<tr>
<td>USERID</td>
<td>42</td>
</tr>
<tr>
<td>USERPROG</td>
<td>349</td>
</tr>
<tr>
<td>UTABL</td>
<td>98</td>
</tr>
<tr>
<td>VALID</td>
<td>5</td>
</tr>
<tr>
<td>VALUE</td>
<td>33</td>
</tr>
<tr>
<td>VB</td>
<td>252</td>
</tr>
<tr>
<td>VERIFY</td>
<td>253</td>
</tr>
<tr>
<td>VHS</td>
<td>15</td>
</tr>
<tr>
<td>VLS</td>
<td>9</td>
</tr>
<tr>
<td>VTAM</td>
<td>254</td>
</tr>
<tr>
<td>WAITING</td>
<td>47</td>
</tr>
<tr>
<td>WARMONLY</td>
<td>268</td>
</tr>
<tr>
<td>YELLOW</td>
<td>255</td>
</tr>
<tr>
<td>YES</td>
<td>1</td>
</tr>
<tr>
<td>3151TERM</td>
<td>409</td>
</tr>
<tr>
<td>3270</td>
<td>256</td>
</tr>
<tr>
<td>3270DBPR</td>
<td>408</td>
</tr>
<tr>
<td>3270DBTM</td>
<td>407</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

3270P  350
3270PRNT  406
3270TERM  405
3275  351
3277  352
3277CM  353
3284  354
3284CM  355
3286  356
3286CM  357
3600  358
3614  359
3650  360
3653  361
3767  362
3767C  363
3767I  364
3770  365
3770B  366
3770C  367
3770I  368
3790  369

EYUDA RESPONSE values in numerical order

This section lists the RESPONSE EYUDAs in numerical order.

<table>
<thead>
<tr>
<th>Value</th>
<th>EYUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>OK</td>
</tr>
<tr>
<td>1025</td>
<td>SCHEDULED</td>
</tr>
<tr>
<td>1026</td>
<td>NOTFOUND</td>
</tr>
<tr>
<td>1027</td>
<td>NODATA</td>
</tr>
<tr>
<td>1028</td>
<td>INVALIDPARM</td>
</tr>
<tr>
<td>1029</td>
<td>FAILED</td>
</tr>
<tr>
<td>1030</td>
<td>ENVIROERROR</td>
</tr>
<tr>
<td>1031</td>
<td>NOTPERMIT</td>
</tr>
</tbody>
</table>
### EYUDA RESPONSE values in alphabetic order

This section lists the RESPONSE EYUDA values in alphabetic order.

<table>
<thead>
<tr>
<th>EYUDA</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1032</td>
<td>BUSY</td>
</tr>
<tr>
<td>1033</td>
<td>SERVERGONE</td>
</tr>
<tr>
<td>1034</td>
<td>NOTAVAILABLE</td>
</tr>
<tr>
<td>1035</td>
<td>VERSIONINV</td>
</tr>
<tr>
<td>1036</td>
<td>INVALIDCMD</td>
</tr>
<tr>
<td>1037</td>
<td>WARNING</td>
</tr>
<tr>
<td>1038</td>
<td>TABLEERROR</td>
</tr>
<tr>
<td>1039</td>
<td>INCOMPATIBLE</td>
</tr>
<tr>
<td>1040</td>
<td>INUSE</td>
</tr>
<tr>
<td>1041</td>
<td>INVALIDATA</td>
</tr>
<tr>
<td>1042</td>
<td>DUPE</td>
</tr>
</tbody>
</table>

### EYUDA REASON values in numerical order

This section lists the REASON EYUDA values in numerical order.

<table>
<thead>
<tr>
<th>EYUDA</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>OK</td>
</tr>
<tr>
<td>1025</td>
<td>SCHEDULED</td>
</tr>
<tr>
<td>1026</td>
<td>NOTFOUND</td>
</tr>
<tr>
<td>1027</td>
<td>NODATA</td>
</tr>
<tr>
<td>1028</td>
<td>INVALIDPARAM</td>
</tr>
<tr>
<td>1029</td>
<td>FAILED</td>
</tr>
<tr>
<td>1030</td>
<td>ENVIRONERROR</td>
</tr>
<tr>
<td>1031</td>
<td>NOTPERMIT</td>
</tr>
<tr>
<td>1032</td>
<td>BUSY</td>
</tr>
<tr>
<td>1033</td>
<td>SERVERGONE</td>
</tr>
<tr>
<td>1034</td>
<td>NOTAVAILABLE</td>
</tr>
<tr>
<td>1035</td>
<td>VERSIONINV</td>
</tr>
<tr>
<td>1036</td>
<td>INVALIDCMD</td>
</tr>
<tr>
<td>1037</td>
<td>WARNING</td>
</tr>
<tr>
<td>1038</td>
<td>TABLEERROR</td>
</tr>
<tr>
<td>1039</td>
<td>INCOMPATIBLE</td>
</tr>
<tr>
<td>1040</td>
<td>INUSE</td>
</tr>
<tr>
<td>1041</td>
<td>INVALIDATA</td>
</tr>
<tr>
<td>1042</td>
<td>DUPE</td>
</tr>
</tbody>
</table>
## Numerical EYUDA REASON values

<table>
<thead>
<tr>
<th>Value</th>
<th>EYUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1280</td>
<td>THREAD</td>
</tr>
<tr>
<td>1281</td>
<td>OBJECT</td>
</tr>
<tr>
<td>1282</td>
<td>CONTEXT</td>
</tr>
<tr>
<td>1283</td>
<td>RESULT</td>
</tr>
<tr>
<td>1284</td>
<td>COUNT</td>
</tr>
<tr>
<td>1285</td>
<td>LENGTH</td>
</tr>
<tr>
<td>1286</td>
<td>FILTER</td>
</tr>
<tr>
<td>1287</td>
<td>NOTFILTER</td>
</tr>
<tr>
<td>1288</td>
<td>FORWARD</td>
</tr>
<tr>
<td>1289</td>
<td>BACKWARD</td>
</tr>
<tr>
<td>1290</td>
<td>POSITION</td>
</tr>
<tr>
<td>1291</td>
<td>DELAY</td>
</tr>
<tr>
<td>1292</td>
<td>NOTIFICATION</td>
</tr>
<tr>
<td>1293</td>
<td>SIGNONParm</td>
</tr>
<tr>
<td>1294</td>
<td>SCOPE</td>
</tr>
<tr>
<td>1295</td>
<td>RESOURCE</td>
</tr>
<tr>
<td>1296</td>
<td>FROM</td>
</tr>
<tr>
<td>1297</td>
<td>TO</td>
</tr>
<tr>
<td>1298</td>
<td>INTO</td>
</tr>
<tr>
<td>1299</td>
<td>CRITERIA</td>
</tr>
<tr>
<td>1300</td>
<td>BY</td>
</tr>
<tr>
<td>1301</td>
<td>ACTION</td>
</tr>
<tr>
<td>1302</td>
<td>ECB</td>
</tr>
<tr>
<td>1303</td>
<td>SENTINEL</td>
</tr>
<tr>
<td>1304</td>
<td>FEEDBACK</td>
</tr>
<tr>
<td>1305</td>
<td>EVENT</td>
</tr>
<tr>
<td>1306</td>
<td>TOKEN</td>
</tr>
<tr>
<td>1307</td>
<td>MODIFY</td>
</tr>
<tr>
<td>1308</td>
<td>VIEW</td>
</tr>
<tr>
<td>1309</td>
<td>FIELDS</td>
</tr>
<tr>
<td>1310</td>
<td>ATTRIBUTE</td>
</tr>
<tr>
<td>1311</td>
<td>FROMMCV</td>
</tr>
<tr>
<td>1312</td>
<td>TOCHAR</td>
</tr>
<tr>
<td>1313</td>
<td>FROMCHAR</td>
</tr>
<tr>
<td>1314</td>
<td>TOCV</td>
</tr>
<tr>
<td>1315</td>
<td>PARM</td>
</tr>
</tbody>
</table>
Numerical EYUDA REASON values

1316  PARMLEN
1317  SUMOPT
1318  TYPE
1319  DATALENGTH
1320  SOLRESOURCE
1321  SOCRESOURCE
1322  SOERESOURCE
1323  MAINTPOINT
1324  SYSNOTACT
1325  SYSLVLBAD
1326  SYSNOTLCL
1327  CICSRELBAD
1328  ARMNOTREG
1329  ARMNOTACT
1330  ARMPOLCHK
1331  ABENDED
1332  CPSMSYSTEM
1333  CPSMVERSION
1334  CPSMAPI
1335  NOTSUPPORTED
1336  NOTVSNCONN
1337  INVALIDATTR
1338  APITASKERR
1339  CPSMSERVER
1340  APITASK
1341  PLEXMGR
1342  REQTIMEOUT
1344  AREATOOSMALL
1345  USRID
1348  VERSION
1352  FILTERMATCH
1353  INVALIDOBJ
1354  INVALIDVER
1355  TASKDATAKEY
1356  INVALIDVERB
1357  NOSTORAGE
1358  NOSERVICE
## Numerical EYUDA REASON values

- **1359** EXCEPTION
- **1360** INVALIDEVT
- **1361** DATAERROR
- **1362** CMAS
- **1363** FIRST
- **1364** NEXT
- **1365** EXPIRED
- **1366** WORKLOAD

## EYUDA REASON values in alphabetic order

This section lists the REASON EYUDAs in alphabetic order.

<table>
<thead>
<tr>
<th>EYUDA</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABENDED</td>
<td>1331</td>
</tr>
<tr>
<td>ACTION</td>
<td>1301</td>
</tr>
<tr>
<td>APITASK</td>
<td>1340</td>
</tr>
<tr>
<td>APITASKERR</td>
<td>1338</td>
</tr>
<tr>
<td>AREATOOSMALL</td>
<td>1344</td>
</tr>
<tr>
<td>ARMNOTACT</td>
<td>1329</td>
</tr>
<tr>
<td>ARMNOTREG</td>
<td>1328</td>
</tr>
<tr>
<td>ARMPOLCHK</td>
<td>1330</td>
</tr>
<tr>
<td>ATTRIBUTE</td>
<td>1310</td>
</tr>
<tr>
<td>BACKWARD</td>
<td>1289</td>
</tr>
<tr>
<td>BY</td>
<td>1300</td>
</tr>
<tr>
<td>CICSRELBAD</td>
<td>1327</td>
</tr>
<tr>
<td>CMAS</td>
<td>1362</td>
</tr>
<tr>
<td>CONTEXT</td>
<td>1282</td>
</tr>
<tr>
<td>COUNT</td>
<td>1284</td>
</tr>
<tr>
<td>CPSMAPI</td>
<td>1334</td>
</tr>
<tr>
<td>CPSMSERVER</td>
<td>1339</td>
</tr>
<tr>
<td>CPSMSYSTEM</td>
<td>1332</td>
</tr>
<tr>
<td>CPSMVERSION</td>
<td>1333</td>
</tr>
<tr>
<td>CRITERIA</td>
<td>1299</td>
</tr>
<tr>
<td>DATAERROR</td>
<td>1361</td>
</tr>
<tr>
<td>DATALENGTH</td>
<td>1319</td>
</tr>
<tr>
<td>DELAY</td>
<td>1291</td>
</tr>
<tr>
<td>ECB</td>
<td>1302</td>
</tr>
<tr>
<td>EVENT</td>
<td>1305</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA REASON values

1359  EXCEPTION
1365  EXPIRED
1304  FEEDBACK
1309  FIELDS
1286  FILTER
1352  FILTERMATCH
1363  FIRST
1288  FORWARD
1296  FROM
1313  FROMCHAR
1311  FROMCV
1298  INTO
1337  INVALIDATTR
1360  INVALIDEVT
1353  INVALIDOBJ
1354  INVALIDVER
1356  INVALIDVERB
1285  LENGTH
1323  MAINTPOINT
1307  MODIFY
1364  NEXT
1358  NOSERVICE
1357  NOSTORAGE
1287  NOTFILTER
1292  NOTIFICATION
1335  NOTSUPPORTED
1336  NOTVSNCONN
1281  OBJECT
1315  PARM
1316  PARMLEN
1341  PLEXMGR
1290  POSITION
1342  REQTIMEOUT
1295  RESOURCE
1283  RESULT
1294  SCOPE
1303  SENTINEL
Alphabetic EYUDA REASON values

1293  SIGNONPARM
1321  SOCRESOURCE
1322  SOERESOURCE
1320  SOLRESOURCE
1317  SUMOPT
1325  SYSLVLBAD
1324  SYSNOTACT
1326  SYSNOTLCL
1355  TASKDATAKEY
1280  THREAD
1297  TO
1312  TOCHAR
1314  TOCV
1306  TOKEN
1318  TYPE
1345  USRID
1348  VERSION
1308  VIEW
1366  WORKLOAD
CICS Transaction Server for z/OS

The above titles are the only unlicensed books available in hardcopy for CICS Transaction Server for z/OS Version 2 Release 1. All the remaining CICS and CICSPlex SM books are supplied in softcopy only in the CICS Information Center, which is distributed on CD-ROM.

CICS books for CICS Transaction Server for z/OS

General

- CICS User's Handbook
  SX33-6116
- CICS Transaction Server for z/OS Glossary
  GC34-5696

Administration

- CICS System Definition Guide
  SC34-5725
- CICS Customization Guide
  SC34-5706
- CICS Resource Definition Guide
  SC34-5722
- CICS Operations and Utilities Guide
  SC34-5717
- CICS Supplied Transactions
  SC34-5724

Programming

- CICS Application Programming Guide
  SC34-5702
- CICS Application Programming Reference
  SC34-5703
- CICS System Programming Reference
  SC34-5726
- CICS Front End Programming Interface User's Guide
  SC34-5710
- CICS C++ OO Class Libraries
  SC34-5705
- CICS Distributed Transaction Programming Guide
  SC34-5708
- CICS Business Transaction Services
  SC34-5704
- Java Applications in CICS
  SC34-5881

Diagnosis

- CICS Problem Determination Guide
  SC33-5713
- CICS Messages and Codes
  GC34-5716
- CICS Diagnosis Reference
  LY33-6097
- CICS Data Areas
  LY33-6096
- CICS Trace Entries
  SC34-5727
- CICS Supplementary Data Areas
  LY33-6096

Communication

- CICS Intercommunication Guide
  SC34-5712
- CICS Family: Interproduct Communication
  SC34-0824
- CICS Family: Communicating from CICS on System/390
  SC34-1697
- CICS External Interfaces Guide
  SC34-5703
- CICS Internet Guide
  SC34-5713

Special topics

- CICS Recovery and Restart Guide
  SC34-5721
- CICS Performance Guide
  SC34-5718
- CICS IMS Database Control Guide
  SC34-5711
- CICS RACF Security Guide
  SC34-5720
CICSplex SM books for CICS Transaction Server for z/OS

General
- CICSPlex SM Concepts and Planning
- CICSPlex SM User Interface Guide
- CICSPlex SM Commands Reference Summary
- CICSPlex SM Web User Interface Guide

Administration and Management
- CICSPlex SM Administration
- CICSPlex SM Operations Views Reference
- CICSPlex SM Monitor Views Reference
- CICSPlex SM Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Managing Business Applications

Programming
- CICSPlex SM Application Programming Guide
- CICSPlex SM Application Programming Reference

Diagnosis
- CICSPlex SM Resource Tables Reference
- CICSPlex SM Messages and Codes
- CICSPlex SM Problem Determination

Other CICS books
- Designing and Programming CICS Applications
- CICS Application Migration Aid Guide
- CICS Family: API Structure
- CICS Family: Client/Server Programming
- CICS Transaction Gateway for OS/390 Administration
- CICS Family: General Information
- CICS 4.1 Sample Applications Guide
- CICS/ESA 3.3 XRF Guide

Note: The CICS Transaction Server for OS/390: Planning for Installation book that was part of the library for CICS Transaction Server for OS/390, Version 1 Release 3, is now merged with the CICS Transaction Server for z/OS Installation Guide. If you have any questions about the CICS Transaction Server for z/OS library, see CICS Transaction Server for z/OS Installation Guide which discusses both hardcopy and softcopy books and the ways that the books can be ordered.

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both hardcopy and BookManager® softcopy versions of a publication are usually in step. However, due to the time required to print and distribute hardcopy books, the BookManager version is more likely to have had last-minute changes made to it before publication.
Subsequent updates will probably be available in softcopy before they are available in hardcopy. This means that at any time from the availability of a release, softcopy versions should be regarded as the most up-to-date.

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

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

A
ADDRESS command
  description 14
  summary of RESPONSE values 139
argument values
  for the command-level interface
    types 1
    using Assembler 4
    using C 3
    using COBOL 2
    using PL/I 3
  for the run-time interface 5
Assembler language programs
  argument values for 4
attributes, resource table
  translating
    with EYUVALUE 9
    with TPARSE 137
    with TRANSLATE 118

C
C programs
  argument values for 3
CANCEL command
  description 17
  summary of RESPONSE values 139
CICS definitions
  creating 27
  removing 102
  updating 125
CICSPlex SM definitions
  creating 27
  removing 102
  updating 125
CICSPlex SM meta-data
  retrieving 53
CICSPlex SM notifications
  canceling 17
  requesting 62
COBOL programs
  argument values for 2
command-level interface
  specifying API commands 1
  specifying argument values 1
command responses
  summary 139
commands
ADDRESS 14
CANCEL 17
CONNECT 19
COPY 23
CREATE 27
DELETE 30
DISCARD 33
DISCONNECT 36
FEEDBACK 38
commands (continued)
FETCH 42
GET 48
GETDEF 53
GROUP 58
LISTEN 62
LOCATE 65
MARK 70
ORDER 74
PERFORM OBJECT 77
PERFORM SET 83
QUALIFY 88
QUERY 91
RECEIVE 94
REFRESH 97
REMOVE 102
SET 105
SPECIFY FILTER 111
SPECIFY VIEW 114
TBUILD 135
TERMINATE 117
TPARSE 137
TRANSLATE 118
UNMARK 121
UPDATE 125
commands, specifying
  using the command-level interface 1
  using the run-time interface 5
CONNECT command
  description 19
  summary of RESPONSE values 139
definitions, CICS
  creating 27
  removing 102
  updating 125
copying result set records 23
CREATE command
  description 27
  summary of RESPONSE values 139
cVDA values, translating 9
D
definitions, CICS
  creating 27
  removing 102
  updating 125
definitions, CICSPlex SM
  creating 27
  removing 102
  updating 125
DELETE command
  description 30
  summary of RESPONSE values 140
deleting result set records 30
DISCARD command
  description 33
  summary of RESPONSE values 140
DISCONNECT command
  description 36
  summary of RESPONSE values 140

E
ECB field
  requesting 14
event control block (ECB)
  requesting 14
EYU9XESV security routine
  options on CONNECT 20
EYUAPI function
  description 130
EYUDA values
  summary of 147
  translating 9
EYUINIT function
  description 131
EYUREAS function
  description 132
EYURESP function
  description 133
EYUTERM function
  description 134
EYUVALUE function
  description 9

F
FEEDBACK command
  description 38
  summary of RESPONSE values 140
FETCH command
  description 42
  summary of RESPONSE values 140
filter
  defining 111
  discarding 33
filter expression
  specifying
    on GET 49
    on PERFORM OBJECT 78
    on SPECIFY FILTER 111
format of commands
  using the command-level interface 1
  using the run-time interface 5

G
GET command
  description 48
  summary of RESPONSE values 141
GETDEF command
  description 53
  summary of RESPONSE values 141
GROUP command
  description 58
  summary of RESPONSE values 142

GROUP command (continued)
  summary of RESPONSE values 141

L
language considerations
  general 9
length options, specifying 10
LISTEN command
  description 62
  summary of RESPONSE values 141
LOCATE command
  description 65
  summary of RESPONSE values 142

M
MARK command
  description 70
  summary of RESPONSE values 142
marking result set records 70
meta-data, CICSPlex SM
  retrieving 53
modification expression
  specifying
    on SET 106
    on UPDATE 126
MVS/ESA restrictions 8

N
notifications, CICSPlex SM
  canceling 17
  requesting 62

O
ORDER command
  description 74
  summary of RESPONSE values 142
order expression
  specifying
    on ORDER 74
    on SPECIFY VIEW 114

P
parameter expression
  specifying
    on CREATE 27
    on GET 49
    on MARK 71
    on PERFORM OBJECT 78
    on PERFORM SET 84
    on REMOVE 102
    on UNMARK 122
    on UPDATE 126
PERFORM OBJECT command
  description 77
  summary of RESPONSE values 142
PERFORM SET command
   description 83
   summary of RESPONSE values 143
PL/I programs
   argument values for 3

Q
QUALIFY command
   description 88
   summary of RESPONSE values 143
QUERY command
   description 91
   summary of RESPONSE values 143

R
REASON option
   description 10
   summary of values 139
RECEIVE command
   description 94
   summary of RESPONSE values 143
REFRESH command
   description 97
   summary of RESPONSE values 143
refreshing result set records 97
REMOVE command
   description 102
   summary of RESPONSE values 144
resource table
   translating attributes
      with EYUVALUE 9
      with TPARSE 137
      with TRANSLATE 118
RESPONSE option
   description 10
   summary of values 139
responses, command
   summary 139
result set
   commands
      COPY 23
      DELETE 30
      DISCARD 33
      FETCH 42
      GET 48
      GROUP 58
      LOCATE 65
      MARK 70
      ORDER 74
      PERFORM OBJECT 77
      PERFORM SET 83
      QUERY 91
      REFRESH 97
      SET 105
      SPECIFY FILTER 111
      SPECIFY VIEW 114
      UNMARK 121
discarding 33
querying 91
result set (continued)
   records
      copying 23
deleting 30
marking 70
refreshing 97
retrieving meta-data 53
REXX commands
   TBUILD 135
   TPARSE 137
REXX functions
   EYUAPI 130
   EYUNIT 131
   EYUREAS 132
   EYURESP 133
   EYUTERM 134
REXX run-time interface
   commands 134
   specifying API commands 5
   specifying argument values 5

S
scope
   changing default 88
   setting default 19
security
   options on CONNECT 20
sentinel field
   requesting 14
SET command
   description 105
   summary of RESPONSE values 144
SPECIFY FILTER command
   description 111
   summary of RESPONSE values 144
SPECIFY VIEW command
   description 114
   summary of RESPONSE values 144
summarized result set
   creating 58
summary expression
   specifying 59
summary options
   specifying 59
syntax diagrams, reading 7

T
TBUILD command
   description 135
TERMINATE command
   description 117
   summary of RESPONSE values 145
TPARSE command
   description 137
TRANSLATE command
   description 118
   summary of RESPONSE values 145
translating
  resource table attributes
  with EYUVALUE 9
  with TPARSE 137
  with TRANSLATE 118

U
UNMARK command
  description 121
  summary of RESPONSE values 145
UPDATE command
  description 125
  summary of RESPONSE values 145

V
view
  building 114
  discarding 33
Notices

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

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

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

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

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

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.
The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

**Sample programs**

This publication contains sample programs. Permission is hereby granted to copy and store the sample programs into a data processing machine and to use the stored copies for internal study and instruction only. No permission is granted to use the sample programs for any other purpose.

**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**

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

- CICS
- CICS/ESA
- CICS/MVS
- CICS/VSE
- CICSPlex
- DB2
- IBM
- IMS
- IMS/ESA
- MVS/ESA
- NetView
- OS/2
- OS/390
- RACF

Other company, product, and service names may be trademarks or service marks of others.
Sending your comments to IBM

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

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

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

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

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

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

• By mail, to this address:
  User Technologies 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 44–1962–842327
  – From within the U.K., use 01962–842327

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