# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>v</td>
</tr>
<tr>
<td>Who this book is for</td>
<td>v</td>
</tr>
<tr>
<td>What you need to know</td>
<td>v</td>
</tr>
<tr>
<td>How to use this book</td>
<td>v</td>
</tr>
<tr>
<td>Notes on terminology</td>
<td>v</td>
</tr>
<tr>
<td>CICS System Connectivity</td>
<td>vi</td>
</tr>
<tr>
<td>Summary of changes</td>
<td>vii</td>
</tr>
<tr>
<td>Changes for CICS Transaction Server for z/OS Version 3 Release 1</td>
<td>vii</td>
</tr>
<tr>
<td>Changes for CICS Transaction Server for z/OS, Version 2 Release 3</td>
<td>vii</td>
</tr>
<tr>
<td>Changes for CICS Transaction Server for z/OS, Version 2 release 2</td>
<td>vii</td>
</tr>
</tbody>
</table>

## Chapter 1. Introduction to the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the command-level interface</td>
<td>1</td>
</tr>
<tr>
<td>Command format</td>
<td>1</td>
</tr>
<tr>
<td>Argument values</td>
<td>1</td>
</tr>
<tr>
<td>Using the run-time interface</td>
<td>5</td>
</tr>
<tr>
<td>Command format</td>
<td>5</td>
</tr>
<tr>
<td>Argument values</td>
<td>5</td>
</tr>
<tr>
<td>Syntax notation used in this book</td>
<td>7</td>
</tr>
<tr>
<td>MVS restrictions</td>
<td>8</td>
</tr>
<tr>
<td>Language considerations</td>
<td>8</td>
</tr>
<tr>
<td>CICS and CICSPlex SM value data areas</td>
<td>8</td>
</tr>
<tr>
<td>Length options</td>
<td>9</td>
</tr>
<tr>
<td>RESPONSE and REASON options</td>
<td>9</td>
</tr>
</tbody>
</table>

## Chapter 2. The API commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>12</td>
</tr>
<tr>
<td>CANCEL</td>
<td>14</td>
</tr>
<tr>
<td>CONNECT</td>
<td>16</td>
</tr>
<tr>
<td>COPY</td>
<td>20</td>
</tr>
<tr>
<td>CREATE</td>
<td>24</td>
</tr>
<tr>
<td>DELETE</td>
<td>27</td>
</tr>
<tr>
<td>DISCARD.</td>
<td>30</td>
</tr>
<tr>
<td>DISCONNECT</td>
<td>33</td>
</tr>
<tr>
<td>FEEDBACK</td>
<td>35</td>
</tr>
<tr>
<td>FETCH.</td>
<td>39</td>
</tr>
<tr>
<td>GET.</td>
<td>46</td>
</tr>
<tr>
<td>GETDEF</td>
<td>52</td>
</tr>
<tr>
<td>GROUP</td>
<td>58</td>
</tr>
<tr>
<td>LISTEN</td>
<td>62</td>
</tr>
<tr>
<td>LOCATE</td>
<td>65</td>
</tr>
<tr>
<td>MARK</td>
<td>70</td>
</tr>
<tr>
<td>ORDER</td>
<td>74</td>
</tr>
<tr>
<td>PERFORM OBJECT</td>
<td>77</td>
</tr>
<tr>
<td>PERFORM SET</td>
<td>83</td>
</tr>
<tr>
<td>QUALIFY</td>
<td>88</td>
</tr>
<tr>
<td>QUERY</td>
<td>91</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>94</td>
</tr>
<tr>
<td>REFRESH</td>
<td>97</td>
</tr>
<tr>
<td>REMOVE</td>
<td>102</td>
</tr>
<tr>
<td>SET</td>
<td>105</td>
</tr>
<tr>
<td>SPECIFY FILTER</td>
<td>111</td>
</tr>
</tbody>
</table>
Preface

This book provides programming information for the 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 CICSPlex (spelled with a lowercase letter ‘p’) means the largest set of CICS systems to be managed by CICSPlex SM as a single entity.

Other terms used in this book are:

<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 can be used to control CICS systems that are directly connected to it.

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

- CICS Transaction Server for z/OS 3.1
- CICS Transaction Server for z/OS 2.3
- CICS Transaction Server for z/OS 2.2
- CICS Transaction Server for OS/390 1.3

You can use this release of CICSPlex SM to control systems running supported releases of CICS 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 from CICS TS Version 2.3 for information on how to do this.

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

<table>
<thead>
<tr>
<th>CICS system</th>
<th>CICSPlex SM component of CICS TS 3.1</th>
<th>CICSPlex SM component of CICS TS 2.3</th>
<th>CICSPlex SM component of CICS TS 2.2</th>
<th>CICSPlex SM component of CICS TS 1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICS TS 3.1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CICS TS 2.3</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CICS TS 2.2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CICS TS 1.3</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TXSeries™ 4.3.0.4</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TXSeries 5.0</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Summary of changes

This book is based on the CICS Transaction Server for z/OS, Version 2 Release 3 edition of the CICSpLex System Manager 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 3 Release 1. Changes made since the last edition are indicated by vertical bars to the left of the change.

Changes for CICS Transaction Server for z/OS Version 3 Release 1

There have been no major changes for this edition.

Changes for CICS Transaction Server for z/OS, Version 2 Release 3

New keywords are added to the GETDEF command to support new resource tables:

- METANAME
- METAPARM
- PARMAVA

See "GETDEF" on page 52 for more information about these changes.

Changes for CICS Transaction Server for z/OS, Version 2 release 2

There has been a change in CICSpLex SM field naming conventions in this release. Data set name fields such as DSNNAME, file name fields such as LOCFILE and REMFILE, and transient data queue name fields such as EXTRATDQ and INTRATDQ are now case-sensitive. When entering data set and file names into the CICSpLex SM interfaces (end user interface, API and the web user interface), ensure that you enter the data in the correct case. In previous releases of CICSpLex SM, the data set names and file names are automatically converted to upper case.

There are no other significant changes for this edition.
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(0310)
  CONTEXT(EYUPLX01) SCOPE(EYUCSG01)
  THREAD(THRDTKN)
  RESPONSE(RESPVAR) REASON(REASVAR)
```

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

Argument values

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

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

  The data you pass can be fullword binary data, fixed or variable length character data, or unspecified. If the data type is unspecified, CICSPlex SM assumes a composite data structure made up of multiple fields of varying data types. The argument can be in one of these forms:
  - Variable name
  - Self-defining term
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

```plaintext
data-ref
Can be replaced by any PL/I data reference that has the correct data type for the argument. The data type can be specified as one of the following:
- Halfword binary — FIXED BIN(15)
- Fullword binary — FIXED BIN(31)
- Character string — CHAR(n) where “n” is the number of bytes.

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

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

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

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

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

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

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

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

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

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

The argument values can be replaced as follows:

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

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

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(0310)''
    , 'THREAD(THDRTKN)''
    , 'RESPONSE(RESPVAR)''
    , 'REASON(REASVAR)')

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

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

data-value
A sending argument used to pass character or binary data from your program to CICSPlex SM.
using the run-time interface

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

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

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

- Character data is not translated; the data is placed into the variable as is.
- Binary data is translated to display format (decimal) and placed into the variable.
- User tokens are not translated; the token value is placed into the variable as is.
- Address values are not translated; the specified storage buffer is placed directly into one or more variables.

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

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

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

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

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

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

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

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

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

**Note:** Each variable (or stem variable) returned by CICSPlex SM contains an entire resource table record. You can use the TPARSE command to break a record into individual fields. For a description of this command, see Chapter 3, “REXX functions and commands,” on page 129.
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:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="symbol" alt="A set of mutually exclusive alternatives, one of which you must code." /></td>
<td>A set of mutually exclusive alternatives, one of which you <em>must</em> code.</td>
</tr>
<tr>
<td><img src="symbol" alt="A set of mutually exclusive alternatives, one of which you may code." /></td>
<td>A set of mutually exclusive alternatives, one of which you <em>may</em> code.</td>
</tr>
<tr>
<td><img src="symbol" alt="A set of alternatives, any number of which you may code." /></td>
<td>A set of alternatives, any number of which you <em>may</em> code.</td>
</tr>
<tr>
<td><img src="symbol" alt="Alternatives where A is the default." /></td>
<td>Alternatives where A is the default.</td>
</tr>
<tr>
<td><img src="symbol" alt="See the separate syntax fragment whose name is shown." /></td>
<td>See the separate syntax fragment whose name is shown.</td>
</tr>
</tbody>
</table>

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 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 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 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.

<table>
<thead>
<tr>
<th>Resource table</th>
<th>Attribute value</th>
<th>CICS Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCTRAN</td>
<td>RESSEC(RESECEXT)</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.
CICS and CICSPlex SM value data areas

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

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

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 149.

---

**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 the maximum limit of all input data areas on an API command is typically 16,124 bytes. When this limit is exceeded the API command fails with a response of INVALIDCMD and a reason of LENGTH.

---

**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
RESPONSE and REASON options

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.

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 141.
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 12
- "CANCEL" on page 14
- "CONNECT" on page 16
- "COPY" on page 20
- "CREATE" on page 24
- "DELETE" on page 27
- "DISCARD" on page 30
- "DISCONNECT" on page 33
- "FEEDBACK" on page 35
- "FETCH" on page 39
- "GET" on page 46
- "GETDEF" on page 52
- "GROUP" on page 58
- "LISTEN" on page 62
- "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
ADDRESS

Provide access to CICSPlex SM storage areas.

ADDRESS

<table>
<thead>
<tr>
<th>ECB(ptr-ref)</th>
<th>SENTINEL(ptr-ref)</th>
<th>THREAD(cpsm-token)</th>
</tr>
</thead>
</table>

RESPONSE(data-ref) — REASON(data-ref)

Description

The ADDRESS command provides access to CICSPlex SM storage areas.
- ADDRESS returns the addresses of two control fields that are associated with each API thread:
  - the event control block (ECB)
  - the sentinel.
- If your program is written in REXX, the ECB and sentinel values are returned as character representations of the hexadecimal addresses. You have to use the REXX STORAGE function to access the storage at those addresses.

Related commands

LISTEN, RECEIVE

Options

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

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

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

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

The sentinel value increases each time an asynchronous request completes. Examples of asynchronous requests include:
- A command is issued with the NOWAIT option
- An event occurs that is named in a LISTEN command.

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

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

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

Cancel the notification request produced by a previous LISTEN command.

\[\text{CANCEL—NOTIFICATION(cpsm-token)—THREAD(cpsm-token)—RESPONSE(data-ref)}\]

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

Description

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

Related commands

LISTEN

Options

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

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

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

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

Conditions

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

OK
The command completed processing successfully.

FAILED
The command failed for one of the following reasons:

ABENDED
Command processing abended.

EXCEPTION
Command processing encountered an exceptional condition.

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

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

NOSTORAGE
The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
CANCEL

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

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

NOTIFICATION
THREATH.

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

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) SIGNONPARM(data-value)

CONNECT(data-value) THREAD(data-ref)
SCOPE(data-value)

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 CICSpesx.

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

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

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

- GET
- PERFORM OBJECT
- PERFORM SET
- REFRESH
- SET.

**SIGNONPARM(data-value)**

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

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

**THREAD(data-ref)**

Names a variable to receive the fullword token that CICSPlex SM assigns to this processing thread.

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

**USER(data-value)**

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

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

**VERSION(data-value)**

Identifies the release of CICSPlex SM resource table data that you want to be available to your program. The VERSION value must be the 4-character number of a valid CICSPlex SM release, such as 0310 for CICS Transaction Server for z/OS, Version 3 Release 1.

**Notes:**

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

- 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.
CONNECT

**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)

- 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 processin thread.

- The COPY command always begins processing with the last record that was fetched, rather than the next one in the result set.
- The target result set can be an existing result set or a new one that is created by this process. If you specify an existing result set as the target, you can either overwrite the existing records or add to them.
- A result set can contain only one record for a given resource. If duplicate records are found during the copy process, the ADD, REPLACE or UPDATE option you specified determines which record is retained.
- To copy selected records from a source result set, you can use:
  - The SPECIFY FILTER command to define a filter for the source result set.
  - The MARK and UNMARK commands to mark records in the source result set.
    - Any marks you place on records in the source result set are not retained when those records are copied to the target result set.
- The relative position of records in the target result set may not be the same as it was in the source result set. The position can be affected by:
  - Deleted records being left in the source result set (when COPY ALL is specified) and other records assuming their position in the target result set.
  - The sort order associated with the target result set, if any. If the target result set does not exist, records are copied in the same order as they appeared in the source result set. If an existing result set is named as the target, records are copied and then sorted according to the sort order that was in effect for that result set.

Related commands

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

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

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

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

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

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

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

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

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

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

- COPY
- GET
- GETDEF
- PERFORM OBJECT.

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

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

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

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

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

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.

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.
CREATE

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 CICSpex 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 CICSpex.

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 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 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 null (that is, zero) values.

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 CICSpex 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 description of the parameters that are valid for a given resource table, see CICSPlex System Manager Resource Tables Reference.
CREATE

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

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

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

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

Conditions

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

OK The command completed processing successfully.

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

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

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

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

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

FAILED
The command failed for one of the following reasons:

ABENDED
Command processing abended.

EXCEPTION
Command processing encountered an exceptional condition.

INVALIDCMD
The command is invalid for the following reason:

LENGTH
The total length of all the options on the command exceeds the maximum limit.

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

CONTEXT
FROM
LENGTH
OBJECT
PARM
PARMLEN
CREATE

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.

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.
### DELETE

Delete resource table records.

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

### Description

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

- The DELETE command always begins processing with the last record that was fetched, rather than the next one in the result set.
- The records you delete are marked as deleted, but they retain their positions in the result set. The remaining records also retain their positions; they are not renumbered. Any API commands that you issue after a DELETE command skip over the deleted records in a result set. One exception is the ORDER command, which sorts all the records in a result set, including deleted records. If you try to issue a command against a deleted record, you receive a RESPONSE value of NODATA.
- To remove deleted records and compress a result set, you can copy the remaining records to a new result set. Use the COPY command with the ALL option to copy all the records in a result set except those that have been deleted.

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

### Related commands

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

### Options

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

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

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

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

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

The \textit{cpsm-token} value that identifies a filter is returned by the \texttt{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 \texttt{MARK} and \texttt{UNMARK} commands.

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

The \textit{cpsm-token} value that identifies a filter is returned by the \texttt{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 \texttt{MARK} and \texttt{UNMARK} commands.

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

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

**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}.

**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 \texttt{CONNECT} command.

**Conditions**

The following is a list of the \texttt{RESPONSE} values that can be returned by the \texttt{DELETE} command. The description of each \texttt{RESPONSE} includes a list of associated \texttt{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 \texttt{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.

INVALID Parm
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.

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.

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

VERSION INVL
A version conflict occurred for one of the following reasons:
NOT SUPPORTED
The version of the application stub program used for this command is not supported.
NOTVSNCONN
The version of the application stub program used for this command is not the same as the version used with the CONNECT command.
**DISCARD**

**DISCARD**

Discard a result set, filter, or view.

\[\text{DISCARD} \rightarrow \text{FILTER}(cpsm\text{-token}) \rightarrow \text{RESULT}(cpsm\text{-token}) \rightarrow \text{VIEW}(data\text{-value}) \rightarrow \text{THREAD}(cpsm\text{-token}) \rightarrow \text{RESPONSE}(data\text{-ref}) \rightarrow \text{REASON}(data\text{-ref})\]

**Description**

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

**Related commands**

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

**Options**

**FILTER**(cpsm-token)

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

**REASON**(data-ref)

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

**RESPONSE**(data-ref)

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

**RESULT**(cpsm-token)

Identifies the API result set to be discarded. The result set can be one produced by any of these commands:

- COPY
- GET
- GETDEF
- GROUP
- PERFORM OBJECT.

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

**THREAD**(cpsm-token)

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

**VIEW**(data-value)

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

**Conditions**

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

BUSY A busy condition occurred for the following reason:
RESULT The result set specified on the RESULT option is being processed
        by another command.

ENVIRONERROR An environment error occurred for one of the following reasons:
NOSERVICE The application stub program could not load the API service
        module.
NOSTORAGE The application stub program could not obtain the necessary
        storage in the address space where the processing thread is
        running.
SOCRESOURCE A required resource that is owned by the CMAS is not available.

FAILED The command failed for one of the following reasons:
ABENDED Command processing abended.
EXCEPTION Command processing encountered an exceptional condition.

INUSE An in use condition occurred for one of the following reasons:
FILTER The specified filter is currently in use and cannot be discarded.
VIEW The specified view is currently in use and cannot be discarded.

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

Check the command description for valid parameter syntax.

NOTAVAILABLE A not available condition occurred for one of the following reasons:
APITASK The API control subtask is not active.
CPSMAPI The CMAS to which the processing thread is connected is not
        available for API processing.

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

VERSIONINVL A version conflict occurred for one of the following reasons:
NOTSUPPORTED The version of the application stub program used for this command
        is not supported.
NOTVSNCONN
The version of the application stub program used for this command is not the same as the version used with the CONNECT command.
DISCONNECT

Disconnects an API processing thread from CICSPlex SM.

```plaintext
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**
DISCONNECT

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:
APITASK
  The API control subtask is not active.
CPSMAPI
  The CMAS to which the processing thread is connected is not available for API processing.
SERVERGONE
  The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:
NOTSUPPORTED
  The version of the application stub program used for this command is not supported.
NOTVSNCONN
  The version of the application stub program used for this command is not the same as the version used with the CONNECT command.
FEEDBACK

Retrieve diagnostic data.

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

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

THREAD(cpsm-token)

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

Description

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

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

Note: If a command that processed a result set returned a RESPONSE value of OK, FEEDBACK records are produced if CICS returns additional information in the EIBRESP2 field.

• 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.
FEEDBACK

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.
FEEDBACK

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:
  COUNT
  INTO
  LENGTH
  RESULT
  THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:
  APITASK
    The API control subtask is not active.
  CPSMAPI
    The CMAS to which the processing thread is connected is not available for API processing.

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

VERSIONINVL
A version conflict occurred for one of the following reasons:
  NOTSUPPORTED
    The version of the application stub program used for this command is not supported.
  NOTVSNCN
    The version of the application stub program used for this command is not the same as the version used with the CONNECT command.
Description

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

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

- 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
FETCH

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.
    • MARKED
    • NOTMARKED
    • FILTER
    • NOTFILTER
  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.

CICSNAME(data-value)
  Specifies a 1- to 8-character specific or generic CICS system name to be used for this operation.
  The CICSNAME option indicates that only those resource table records that originate from CICS systems that match the specified name pattern should be considered for retrieval. When CICSNAME is specified in conjunction with FILTER or NOTFILTER, only records which meet the FILTER or NOTFILTER requirements and also match the CICSNAME pattern will be considered. The number of records actually retrieved is determined by the COUNT option.
  When you specify CICSNAME, the result set named on the RESULT option must not be a summarized result set and must contain resource table records that have an EYU_CICSNAME attribute. If the result set specified by RESULT contains summarized records or resource table records that do not have an EYU_CICSNAME attribute, you receive an INVALIDPARM response for the CICSNAME option.

COUNT(data-ref)
  Specifies the number of resource table records to be processed.
  The COUNT option applies to the result set named in the RESULT option.
  When you also specify the DETAIL option, COUNT provides the number of summary records in the summarized result set in RESULT for which source records are returned. The OBJSTAT table for each summary record contains the number of source records that will be returned for that record if the DETAIL option is specified.
If you do not specify the COUNT option, the default is one.

If the COUNT option is specified, COUNT contains the number of records processed. In most cases this is also the number of records returned. However, if you also specify the DETAIL option, all source records associated with the requested number of summary record are retrieved. This is normally greater than the number specified in the COUNT option.

If you are retrieving multiple records, they are placed one after another in the INTO buffer. The INTO buffer should 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 as follows:

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.

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

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

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

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

When you specify DETAIL, the result set named in 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 options along with DETAIL to select which summary record you want to expand. The FORWARD and BACKWARD options also control the direction in which records are retrieve from the source result set.

By default, all the source records associated with the summary record or records are retrieved. However, you can use the 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. If the APPLYSUMMARY option is specified, FILTER, NOTFILTER,
FETCH

MARKED, and NOTMARKED options are applied to records in the summary result set rather than to the source records.

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)**.

**Note:** When the POSITION option is used with the DETAIL option to retrieve source records for a specific summarized result set record, the value of the COUNT option is forced to one (1). In this case, the value returned by the COUNT option is the number of source records summarized in the specified result set record.

**REASON**(data-ref)

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

**RESPONSE**(data-ref)

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

**RESULT**(cpsm-token)

Identifies the API result set to be processed by this operation. The result set can be one produced by any of these commands:

- COPY
- GET
- GETDEF
- GROUP
- PERFORM OBJECT.

**STATUS**

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

**THREAD**(cpsm-token)

Identifies the API thread to be used for this operation. The cPSM-token value that identifies a thread is returned by the CONNECT command.

**Conditions**

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

**OK**  The command completed processing successfully.

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

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

WARNING
The command completed processing with a warning, for the following reason:
AREATOOSMALL
The INTO buffer is not long enough to hold the number of records requested and available.

BUSY
A busy condition occurred for the following reason:

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

ENVIRONERROR
An environment error occurred for one of the following reasons:
NOSERVICE
The application stub program could not load the API service module.
NOSTORAGE
The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
SOCRESOURCES
A required resource that is owned by the CMAS is not available.
SOLRESOURCES
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
CICSNAME

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

Return a result set containing selected resource table records.

```
GET—OBJECT(data-value) COUNT(data-ref)

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

PARM(data-area) PARMLEN(data-value) NOWAIT
TOK(data-value)

CONTEXT(data-value) SCOPE(data-value)
RESULT(cpsm-token)

THREAD(cpsm-token) RESPONSE(data-ref) REASON(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(RESSECEXT)</td>
<td>CICS/MVS</td>
</tr>
</tbody>
</table>

The values returned by CICSPlex SM for: LOCTRAN RESSEC(RESSECEXT) do not match the CVDA values returned by CICS. The values returned by CICS conflict with CVDA values in other CICS environments. In order to retain the attributes’ uniqueness, CICSPlex SM adds 9000 to the values returned by CICS.

Related commands

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

Options

**CONTEXT(data-value)**

Identifies the context for this command. The context must be the 1- to 8-character name of a CMAS or CICSpex.

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.
GET

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 preselecting resource table records.

For details on how to use a parameter expression with the GET command, see CICSPlex System Manager Application Programming Guide For a description of the parameters that can be specified 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 can be one produced by any of these commands:
GET

- COPY
- GET
- GROUP
- PERFORM OBJECT.

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

If this field is:
- Set to binary zero (in COBOL, C, PL/I or Assembler)
- An uninitialized variable (in REXX).

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

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

If the current context (as set by this command or a previous CONNECT or QUALIFY command) is a CICSp lex 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 CICSp lex itself
- A CICS system or CICS system group within the CICSp lex
- A logical scope, as defined in a CICSPlex SM resource description (RESDESC).

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

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

STATE(cpsm-token)
Returns active or inactive topology objects. The default is to return both.

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.
        If only some of the CMASs or MASs did not respond, the GET can
        still yield a valid result. COUNT might be greater than zero and
        RESULT might be non-zero. Such a result set contains data from
        those CMASs and MASs that did respond.
SOCRESOURCE  A required resource that is owned by the CMAS is not available, or
        the maximum number of result set, view or filter tokens has been
        reached.
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.
INVALIDCMD  The command is invalid for one of the following reasons:
FILTER  The filter expression passed on the operation is too large or
        complex.
LENGTH  The total length of all the options on the command exceeds the
        maximum limit.
INVALIDPARM  An invalid parameter was detected. The parameter that is invalid is returned
        as the reason value:
        CONTEXT
        CRITERIA
        FILTER
        LENGTH
        OBJECT
GET

PARM
PARMLEN
RESULT
SCOPE
TOKEN
THREAD.

Check the command description for valid parameter syntax.

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

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

MAXRECORDS
The number of records added to the result set by a MAS would have exceeded the MAXHISTRECS value for that MAS. Records within the MAXHISTRECS limit have been added to the result set. Modify the FILTER or PARM parameter values to reduce the number of records the MAS should add to the result set.
GETDEF

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

GETDEF

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

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 "Object" on page 53.
- 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

- ACTION(data-value)
  The 12-character name of an action against the resource table for which CPSM MetaData records are to be retrieved.
The 12-character name of a parameter to an action against the resource table for which CPSM MetaData records are to be retrieved, as it appears in the API parameter string.

ADD

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

ATTRIBUTE(data-value)

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

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

COUNT(data-ref)

Names a variable to receive the number of resource table records in the target result set after this operation is complete.

OBJECT(data-value)

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

OBJECT

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

- ACTION is ignored.
- ATTRIBUTE is ignored.
- ACTIONPARM 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:

- ACTION is normally omitted. If present, it may specify the name of an action or *.
- ATTRIBUTE is ignored.
- ACTIONPARM 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:
- ACTION is ignored.
- ATTRIBUTE can be a specific attribute name or * for all attributes in
  the resource table.
- ACTIONPARM is ignored.
- RESOURCE must be a specific resource table name; a value of * is
  not allowed.

**ATTR**
One record is returned for each attribute of the resource table specified
on the RESOURCE option. Each record provides complete information
about the attribute.

Related options and restrictions include:
- ATTRIBUTE can be a specific attribute name or * for all attributes in
  the resource table.
- RESOURCE must be a specific resource table name; a value of * is
  not allowed.

**ATTRAVA**
One record is returned for each of the EYUDA or CVDA values that are
valid for the specified attribute.

Related options and restrictions include:
- ATTRIBUTE must be the name of a specific attribute that has a data
type of EYUDA, CVDAS, or CVDAT.
- RESOURCE must be a specific resource table name; a value of * is
  not allowed.

**Note:** The AVAAVAIL attribute of the ATTR internal resource table
indicates whether an AVA list is available for a given attribute.

**METANAME**
One record is returned for each CVDA and EYUDA. The RESOURCE,
ATTRIBUTE, ACTION and ACTIONPARM keywords are ignored.

**METAPARM**
One record is returned for every parameter for the specified
RESOURCE and ACTION.
- ACTION must be a specific action name; a value of * is not allowed.
- ACTIONPARM is ignored.
- RESOURCE must be a specific table; a value of * is not allowed.

**PARMAVA**
One record is returned for the specified RESOURCE, ACTION, and
ACTIONPARM.
- ACTION must be a specific action name; a value of * is not allowed.
- ACTIONPARM must be a specific parameter name; a value of * is
  not allowed.
- RESOURCE must be a specific table; a value of * is not allowed.

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

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

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

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

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

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

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

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

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

If this field is:
- Set to binary zero (in COBOL, C, PL/I or Assembler)
- An uninitialized variable (in REXX).

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

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

Conditions

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

OK The command completed processing successfully.

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

BUSY A busy condition occurred for the following reason:
GETDEF

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

ENVIRONERROR
   An environment error occurred for one of the following reasons:
   
NOSERVICE
   The application stub program could not load the API service module.
   
NOSTORAGE
   The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
   
SOCRESOURCE
   A required resource that is owned by the CMAS is not available.

FAILED
   The command failed for one of the following reasons:
   
ABENDED
   Command processing abended.
   
EXCEPTION
   Command processing encountered an exceptional condition.

INCOMPATIBLE
   An incompatible condition occurred for the following reason:
   
INVALIDOBJ
   The target result set specified on the RESULT option is not compatible with the output of this command. The result set must contain the same type of meta-data (as specified on the OBJECT option) as the command produces.

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

GROUP

Return a summarized result set.

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

ALL

FILTER(cpsm-token)  COUNT(data-ref)

MARKED

NOTFILTER(cpsm-token)

NOTMARKED

SUMOPT(data-area)—LENGTH(data-value)—THRED(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\textit{(data-ref)}
Names a variable to receive the number of resource table records in the target result set after this operation is complete.

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

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

FROM\textit{(cpsm-token)}
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}

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

LENGTH\textit{(data-value)}
A fullword value that specifies the length of the SUMOPT buffer.

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

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

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

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

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

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

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

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

For details on how to form a summary expression, see 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.

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

FAILED The command failed for one of the following reasons:
GROUP

ABENDED
Command processing abended.

EXCEPTION
Command processing encountered an exceptional condition.

INVALIDCMD
The command is invalid for the following reason:

LENGTH
The total length of all the options on the command exceeds the maximum limit.

INVALIDPARAM
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:

API TASK
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.

\[\text{LISTEN—EVENT(data-value)—NOTIFICATION(data-ref)}\]

\[\text{FILTER(cpsm-token)}\]
\[\text{NOTFILTER(cpsm-token)}\]
\[\text{TOKEN(data-value)}\]
\[\text{CONTEXT(data-value)}\]
\[\text{THREAD(cpsm-token)—RESPONSE(data-ref)—REASON(data-ref)}\]

Description

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

- 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 CICSpex 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 CICSpex.

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 CICSpex SM resource tables by type, see CICSpex 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 CICSpex. For example, you could create a filter like this:

\[\text{PLEXNAME=EYUPLX01.}\]

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

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

\textbf{NOTFILTER}(\textit{cpsm-token})

Identifies a filter to be used for this operation. The \textit{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 \textit{SPECIFY FILTER} command.

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

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

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

\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{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 \textit{CONNECT} command.

\textbf{TOKEN}(\textit{data-value})

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

\textbf{Conditions}

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

\textbf{OK} The command completed processing successfully.

\textbf{ENVIRONERROR}

An environment error occurred for one of the following reasons:

\textbf{NOSERVICE}

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

\textbf{NOSTORAGE}

The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

\textbf{SOCRESOURCE}

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

\textbf{FAILED}

The command failed for one of the following reasons:

\textbf{ABENDED}

Command processing abended.

\textbf{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.

```plaintext
LOCATE pointer_group
     RESULT(cpsm-token) THREAD(cpsm-token)
     RESPONSE(data-ref) REASON(data-ref)
```

**pointer_group**

- `POSITION_group`
- `FILTER_group`

**POSITION_group**

- `TOP`
- `BOTTOM`
- `POSITION(data-value)`
- `FORWARD(data-value)`
- `BACKWARD(data-value)`

**FILTER_group**

- `FILTER(cpsm-token)`
- `MARKED`
- `NOTFILTER(cpsm-token)`
- `NOTMARKED`
- `FIRST`
- `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.

- 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.

Chapter 2. The API commands 65
LOCATE

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 last resource table record in the result set.

FILTER(cpsm-token)
Identifies a filter to be used for this operation.
The FILTER option positions the record pointer to a resource table record that meets the specified filter criteria. The FIRST, LAST, NEXT, or PREV option determines where in the result set the search begins and in what direction it continues.
The cpsm-token value that identifies a filter is returned by the SPECIFY FILTER command.

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

FORWARD(data-value)
Moves the record pointer forward by the specified number of resource table records.
If the pointer reaches the bottom of the result set, it remains positioned on the last resource table record. The pointer does not continue moving forward to the top of the result set.

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

MARKED
Positions the record pointer to a resource table record that is marked. The FIRST, LAST, NEXT, or PREV option determines where in the result set the search begins and in what direction it continues.
You can mark resource table records by using the MARK and UNMARK commands.

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

NOTFILTER(cpsm-token)
Identifies a filter to be used for this operation.
The NOTFILTER option positions the record pointer to a resource table record that does not meet the specified filter criteria. The FIRST, LAST, NEXT, or PREV option determines where in the result set the search begins and in what direction it continues.

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

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

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

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

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

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

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

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

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

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

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

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

**Conditions**

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

**OK** The command completed processing successfully.
LOCATE

NODATA
No records were found that matched the specified search criteria, for one of the following reasons:
  BACKWARD
  There are no more records that satisfy the search criteria in the backward direction.
  FORWARD
  There are no more records that satisfy the search criteria in the forward direction.

BUSY
A busy condition occurred for the following reason:
  RESULT
  The result set specified on the RESULT option is being processed by another command.

ENVIRONERROR
An environment error occurred for one of the following reasons:
  NOSERVICE
  The application stub program could not load the API service module.
  NOSTORAGE
  The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
  SOCRESOURCE
  A required resource that is owned by the CMAS is not available.
  SOLRESOURCE
  A required resource that is locally owned (that is, owned by the address space where the processing thread is running) is not available.

FAILED
The command failed for one of the following reasons:
  ABENDED
  Command processing abended.
  EXCEPTION
  Command processing encountered an exceptional condition.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:
  BACKWARD
  FILTER
  FORWARD
  NOTFILTER
  POSITION
  RESULT
  THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:
  APITASK
  The API control subtask is not active.
  CPSMAPI
  The CMAS to which the processing thread was trying to connect is not available for API processing.
SERVERGONE
The CMAS to which the processing thread was connected is no longer active.

VERSIONINVL
A version conflict occurred for one of the following reasons:
NOTSUPPORTED
The version of the application stub program used for this command is not supported.
NOTVSNCONN
The version of the application stub program used for this command is not the same as the version used with the CONNECT command.
MARK

Mark selected resource table records in a result set.

```
MARK records_group COUNT(data-ref)
INTO(data-area) LENGTH(data-ref) RESULT(cpsm-token)
RESET
THREAD(cpsm-token) RESPONSE(data-ref) REASON(data-ref)
```

**Description**

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

- The MARK command always begins processing with the last record that was fetched, rather than the next one in the result set.
- Any resource table records that you marked in the result set previously remain marked unless you use the RESET option.

**Related commands**

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

**Options**

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

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

- **CURRENT**
  Marks only the current resource table record.

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

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

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

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.

**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.
MARK

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

ORDER

Sort the resource table records in a result set.

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.
ORDER

**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.

**INVALIDCMD**
The command is invalid for the following reason:

**LENGTH**
The total length of all the options on the command exceeds the maximum limit.

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

LENGTH
RESULT
THREAD.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:
APITASK
The API control subtask is not active.
CPSMAPI
The CMAS to which the processing thread is connected is not available for API processing.

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

VERSIONINVL
A version conflict occurred for one of the following reasons:
NOTSUPPORTED
The version of the application stub program used for this command is not supported.
NOTVSNCONN
The version of the application stub program used for this command is not the same as the version used with the CONNECT command.
PERFORM OBJECT

Perform an action on one or more resources.

```
PERFORM OBJECT(data-value) ACTION(data-value) COUNT(data-ref)
```

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 CICSpex.

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.
PERFORM OBJECT

**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.

**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**.

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

**OBJECT**(data-value)
Identifies the resource table against which the action is to be performed. This value must be the 8-character name of a valid resource table.

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

For details on how to use a parameter expression with the PERFORM OBJECT command, see **CICSPlex 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.
PERFORM OBJECT

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 CICSpex. 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, 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; CICSpex SM makes no use of the value. The token is returned by the RECEIVE command when this PERFORM OBJECT request is complete.

Conditions

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

OK The command completed processing successfully.
PERFORM OBJECT

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:
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.

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

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 DATA
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.

INVALID CMD
The command is invalid for one of the following reasons:
FILTER
The filter expression passed on the operation is too large or
complex.

LENGTH
The total length of all the options on the command exceeds the
maximum limit.

INVALID PARM
An invalid parameter was detected. The parameter that is invalid is returned
as the reason value:
ACTION
CONTEXT
CRITERIA
FILTER
PERFORM OBJECT

LENGTH
OBJECT
PARM
PARMLEN
RESULT
SCOPE
THREAD
TOKEN.

Check the command description for valid parameter syntax.

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

APITASK
The API control subtask is not active.

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

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

MAINTPOINT
The maintenance point for the current context is not available.

PLEXMGR
The CMAS to which the processing thread is currently connected does not participate in managing the specified CICSpelix and no other CMAS is available that does manage the CICSpelix.

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.
PERFORM OBJECT

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:

ACTION
The specified action is not supported for the version used with the CONNECT command.

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.

\[ \text{PERFORM SET ACTION}(\text{data-value}) \]

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

\[ \text{NOWAIT} \]

\[ \text{TOKEN}(\text{data-value}) \]

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

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

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

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

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.
PERFORM SET

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
don't respond.

SOCRESOURCE 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.

INVALIDCMD
The command is invalid for the following reason:

LENGTH
The total length of all the options on the command exceeds the maximum limit.

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

ACTION
FILTER
NOTFILTER
PARM
PARMLEN
POSITION
RESULT
THREAD
TOKEN.

Check the command description for valid parameter syntax.

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

APITASK
The API control subtask is not active.

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

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

MAINTPOINT
The maintenance point for the current context is not available.

PLEXMGR The CMAS to which the processing thread is currently connected does not participate in managing the specified CICSPlex and no other CMAS is available that does manage the CICSPlex.

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

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.
PERFORM SET

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

Define the CICSPlex SM context and scope.

\[
\text{QUALIFY} - \text{CONTEXT(data-value)} - \text{SCOPE(data-value)} - \text{THREAD(cpsm-token)} - \text{RESPONSE(data-ref)} - \text{REASON(data-ref)}
\]

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
QUALIFY

- 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.

**ENVIROMERROR**
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.

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

**FAILED**
The command failed for one of the following reasons:

**ABENDED**
Command processing abended.

**EXCEPTION**
Command processing encountered an exceptional condition.

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

**CONTEXT**

**SCOPE**

**THREAD**.

Check the command description for valid parameter syntax.

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

**APITASK**
The API control subtask is not active.

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

**PLEXMGR**
The CMAS to which the processing thread is currently connected does not participate in managing the specified CICSpex and no other CMAS is available that does manage the CICSpex.

**SERVERGONE**
The CMAS to which the processing thread was connected is no longer active.
VERSIONINVL
A version conflict occurred for one of the following reasons:

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

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

Retrieve information about a result set and the resource table records it contains.

```
QUERY RESULT(cpsm-token)
  COUNT(data-ref)
  DATALENGTH(data-ref)
  OBJECT(data-ref)
  TYPE(data-ref)
  CONTEXT(data-ref)
  SCOPE(data-ref)
  THREAD(cpsm-token)
  RESPONSE(data-ref)
  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.

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

**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:

**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:

**ABENED**
Command processing abended.

**EXCEPTION**
Command processing encountered an exceptional condition.
INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:
  CONTEXT
  DATALENGTH
  OBJECT
  RESULT
  THREAD
  TYPE.

Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:
  APITASK
    The API control subtask is not active.
  CPSMAPI
    The CMAS to which the processing thread is connected is not available for API processing.

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

VERSIONINVL
A version conflict occurred for one of the following reasons:
  NOTSUPPORTED
    The version of the application stub program used for this command is not supported.
  NOTVSNCONN
    The version of the application stub program used for this command is not the same as the version used with the CONNECT command.
RECEIVE

RECEIVE

Receive the output from completed asynchronous requests.

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

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

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

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

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

\text{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.

DELAY must specify a non-zero value. If you want to make sure that your program never enters a wait, use the IMMEDIATE option instead of DELAY.

\text{IMMEDIATE}

Returns control to the processing thread immediately, whether or not any output is available.
RECEIVE

**INTO**(data-area)
Identifies a buffer to receive asynchronous output, if any is available for this thread. This buffer must be long enough to hold all the output being received.

The output returned can be:
- A resource table record representing an event named in a previous LISTEN command
- An ASYNCREQ resource table record representing completion of an asynchronous GET, PERFORM, REFRESH, or SET request.

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

After the operation is complete, this field is set to the actual length of the data returned in the INTO buffer. If the operation cannot complete because the buffer is not long enough, this field is set to the length that is required.

**OBJECT**(data-ref)
Names a variable to receive a resource table name, if output is available for this thread.

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

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

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

**WAIT**
Waits until asynchronous output becomes available before returning control to the processing thread.

**Note:** The WAIT option waits indefinitely for asynchronous output. Be sure to verify that there are completed asynchronous requests outstanding by issuing the ADDRESS command before you issue RECEIVE.

**TOKEN**(data-ref)
Names a variable to receive the user-defined token associated with the asynchronous output. This value is the token you defined on the GET, LISTEN, PERFORM, REFRESH or SET command that produced the output.

**Conditions**

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

**OK** The command completed processing successfully.

**NODATA** There was no data to receive.

**WARNING**
The command completed processing with a warning, for the following reason:
AREATOOSMALL
The INTO buffer is not long enough to hold the number of records requested and available.

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

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

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

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

FAILED
The command failed for one of the following reasons:

ABENDED
Command processing abended.

EXCEPTION
Command processing encountered an exceptional condition.

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

DELAY
INTO
LENGTH
OBJECT
THREAD
TOKEN.

Check the command description for valid parameter syntax.

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

APITASK
The API control subtask is not active.

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

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

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

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

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

Refreshes the data for resource table records.

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

  COUNT(data-value)

  FORWARD

  BACKWARD

  NOWAIT

  TOKEN(data-value)

  RESULT(cpsm-token)

  THREAD(cpsm-token)

  RESPONSE(data-ref)

  REASON(data-ref)

Description

- This command refreshes the data for some or all of the resource table records in a result set.
- For the MAS resource table, REFRESH provides data only if the MAS was active when the result set was last built.

Related commands

COPY, GET, LOCATE, MARK, PERFORM OBJECT, SPECIFY FILTER

Options

ALL

Refreshes all the resource table records in the result set. When you specify ALL:

- The COUNT option is ignored.
- Any records that have been deleted are removed from the result set. Any positions previously held by deleted records are filled in and the remaining records are renumbered. Therefore, the relative position of a given record in a result set may be different after a refresh.

BACKWARD

Refreshes the previous resource table record and continues in a backward direction through the result set refreshing as many records as the COUNT option specifies.

Note: If the record pointer is at the bottom of the result set, using BACKWARD refreshes the current record (which is the last record) and then continues on to previous records.

COUNT(data-value)

Specifies the number of resource table records to be refreshed. If you do not specify the COUNT option, only one record is refreshed.

If you do not specify the FORWARD or BACKWARD option, the refresh process moves in a forward direction through the result set.
REFRESH

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.

VERSIONINVLF
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

**Description**

This command removes a CICSPlex SM or CICS definition from the data repository. For definitions that have a CICSpelix 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 CICSpelix.

**Related commands**

CREATE, UPDATE

**Options**

**CONTEXT**(data-value)

Identifies the context for this command. The context must be the 1- to 8-character name of a CMAS or CICSpelix.

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 CICSpelix SM resource tables by type, see the CICSpelix 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 the CICSpelix System Manager Application Programming Guide. For a description of the parameters that are valid for a given resource table, see the CICSpelix 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.
REMOVE

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.

**SOCRESOURCEx**
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.

**INVALIDCMD**
The command is invalid for the following reason:

**LENGTH**
The total length of all the options on the command exceeds the maximum limit.

**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:
REMOVE

APITASK
The API control subtask is not active.

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

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

MAINTPOINT
The maintenance point for the current context is not available.

NOTPERMIT
A not permitted condition occurred for the following reason:

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

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

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

DATAERROR
The value associated with one or more resource table attributes is invalid. This error can occur if the resource table is missing required attributes, contains one or more conflicting attributes, or does not exist. Use the FEEDBACK command to retrieve additional data about this error.

INVALIDATTR
One of the resource table attributes is invalid.

INVALIDVER
The specified version of the resource table is not supported by CICSPlex SM.

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

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

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

Modify the attributes of one or more resources.

```
SET MODIFY(data-area) LENGTH(data-value) ALL
```

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

```
COUNT(data-value) FORWARD BACKWARD NOREFRESH
```

- **NOWAIT**
- **RESULT(cpsm-token) THRESH(cpsm-token)**

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

**Description**

This command modifies the attributes of one or more resources as represented by resource table records in an existing result set.

- The SET command is valid only for CICS Resource and some CPSM Manager resource tables.
- If the context and scope in effect when you issue a SET command include CICS systems that do not support the requested modification, the request is ignored for those CICS systems.

**Related commands**

COPY, GET, GROUP, LOCATE, MARK, PERFORM OBJECT, PERFORM SET, SPECIFY FILTER

**Options**

- **ALL**
  
  Modifies all the resource table records in the result set. When you specify ALL, the COUNT option is ignored.

- **BACKWARD**
  
  Modifies the previous resource table record and continues in a backward direction through the result set modifying as many records as the COUNT option specifies.

  **Note:** If the record pointer is at the bottom of the result set, using BACKWARD modifies the current record (which is the last record) and then continues on to previous records.

- **COUNT**(data-value)
  
  Specifies the number of resource table records to be modified. If you do not specify the COUNT option, only one record is refreshed.
If you do not specify the FORWARD or BACKWARD option, the modification process moves in a forward direction through the result set.

**CURRENT**
Modifies only the current resource table record. When you specify CURRENT, the COUNT option is ignored.

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

The number of records that are actually modified is determined by the COUNT option. If you do not specify the COUNT option, only the first record that meets the filter criteria is modified.

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

**FORWARD**
Modifies the current resource table record and continues in a forward direction through the result set modifying as many records as the COUNT option specifies.

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

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

**MARKED**
Indicates that only those resource table records that are marked in the result set should be considered for modification.

The number of records that are actually modified is determined by the COUNT option. If you do not specify the COUNT option, only the first record that is marked is modified.

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

**MODIFY**(data-area)
Identifies a buffer containing the modification expression to be used in modifying the resource table records.

For details on how to form a modification expression, see CICSPlex System Manager Application Programming Guide.

**NOREFRESH**
Specifies that the resource table records in the source result set should not be refreshed. The records reflect the status of the resources before the SET command was processed.

If you do not specify the NOREFRESH option, the resource table records are refreshed to reflect the resource status after this operation is complete.

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

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 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 MARK and UNMARK commands.

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

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

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

\textbf{POSITION}(\textit{data-value})
Modifies the \textit{nth} resource table record in the result set. When you specify POSITION, the COUNT option is ignored.

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

For example, to modify the fifth resource table record in a result set, you would specify \textbf{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:

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

\textbf{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.
SET

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.

SOCRESOUCRE 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.

INVALIDCMD The command is invalid for the following reason:

LENGTH The total length of all the options on the command exceeds the
maximum limit.
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.
SET

INVALIDVER
The specified version of the resource table is not supported by CICSplex SM.

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

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

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

**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.

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

**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.

**INVALIDCMD**
The command is invalid for one of the following reasons:

- **FILTER**
  The filter expression passed on the operation is too large or complex.

- **LENGTH**
  The total length of all the options on the command exceeds the maximum limit.

**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.

```plaintext
SPECIFY VIEW(data-value) — FIELDS(data-area) — LENGTH(data-value)

SPECIFY VIEW(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](#).
- If and when you migrate to a later version of CICSPlex SM, you should check that any new resource tables do not have the same names as any customized views. For more details, see [CICSPlex System Manager Application Programming Guide](#).

**Related commands**

DISCARD, GET

**Options**

**FIELDS(data-area)**

Identifies a buffer containing the order expression to be used for this operation.

For details on how to use an order expression with the `SPECIFY VIEW` command, see [CICSPlex 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 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](#).
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.
SOCRESoure 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.

INVALIDCMD The command is invalid for the following reason:

LENGTH The total length of all the options on the command exceeds the maximum limit.

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
SPECIFY VIEW

VIEW.
Check the command description for valid parameter syntax.

NOTAVAILABLE
A not available condition occurred for one of the following reasons:
APITASK
  The API control subtask is not active.
CPSMAPI
  The CMAS to which the processing thread is connected is not available for API processing.

NOTFOUND
A not found condition occurred for the following reason:
ATTRIBUTE
  An attribute specified in the FIELDS buffer was not found for the specified resource table.

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

TABLEERROR
A resource table record is invalid for one of the following reasons:
DATAERROR
  The value associated with one or more resource table attributes is invalid. This error can occur if the resource table is missing required attributes, contains one or more conflicting attributes, or does not exist. Use the FEEDBACK command to retrieve additional data about this error.
INVALIDVER
  The specified version of the resource table is not supported by CICSPlex SM.

VERSIONINVL
A version conflict occurred for one of the following reasons:
NOTSUPPORTED
  The version of the application stub program used for this command is not supported.
NOTVSNCONN
  The version of the application stub program used for this command is not the same as the version used with the CONNECT command.
TERMINATE

Terminate all API processing on all active threads.

TERMINATE—RESPONSE(data-ref)—REASON(data-ref)

Description

This command terminates all API processing on all active threads created by the CICS or MVS 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.
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 129.

- 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.
TRANSLATE

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

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

TOCHAR(*data-ref*)
Names a variable to receive the result of translating an internal coded value to the 1- to 12-character value for the specified attribute.

TOCV(*data-ref*)
Names a variable to receive the result of translating a character value to the 4-byte internal coded value for the specified attribute.

Conditions

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

**OK**
The command completed processing successfully.

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

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

- **NOSTORAGE**
  The application stub program could not obtain the necessary storage in the address space where the processing thread is running.

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

**FAILED**
The command failed for one of the following reasons:

- **ABENDED**
  Command processing abended.

- **EXCEPTION**
  Command processing encountered an exceptional condition.

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

- **ATTRIBUTE**
- **FROMCHAR**
- **FROMCV**
- **OBJECT**
- **THREAD**
- **TOCHAR**
- **TOCV**.

Check the command description for valid parameter syntax.

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

- **APITASK**
  The API control subtask is not active.
**TRANSLATE**

**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.

```plaintext
UNMARK 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 removes the marks placed on resource table records by a previous `MARK` command. The `UNMARK` command always begins processing with the last record that was fetched, rather than the next one in the result set.

### Related commands

`LOCATE`, `MARK`

### Options

**ALL**
- Removes the marks from all resource table records in the result set.

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

**CURRENT**
- Removes the mark from only the current resource table record.

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

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

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

  This buffer must be long enough to hold the maximum number of record numbers that could result from your `UNMARK` request (in the event that none of these record numbers are present in your result set).
UNMARK

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.

INVALIDCMD
The command is invalid for the following reason:

LENGTH
The total length of all the options on the command exceeds the maximum limit.

INVALIDPARM
An invalid parameter was detected. The parameter that is invalid is returned as the reason value:
COUNT
FILTER
INTO
LENGTH
NOTFILTER
PARM
PARMLEN
RESULT
THREAD.
Check the command description for valid parameter syntax.

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

APITASK
The API control subtask is not active.

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

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

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

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

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

Update an existing CICSPlex SM or CICS definition.

\[ UPDATE \quad OBJECT(data-value) \]

\[ FROM(data-area) \]
\[ PARM(data-area) \quad PARMLEN(data-value) \]
\[ RESULT(cpsm-token) \quad MODIFY(data-area) \]

\[ LENGTH(data-value) \]
\[ CONTEXT(data-value) \quad THREAD(cpsm-token) \]

\[ RESPONSE(data-ref) \quad REASON(data-ref) \]

Description

This command updates an existing CICSPlex SM or CICS definition according to the attribute values you specify.

- The updated definition replaces the existing definition in the CICSPlex SM data repository.
- For definitions that have a CICSplex as their context (such as workload management or real-time analysis definitions), the definition is also updated in the data repositories of all CMASs involved in managing the CICSplex.

Related commands

CREATE, REMOVE

Options

**CONTEXT(data-value)**

Identifies the context for this command. The context must be the 1- to 8-character name of a CMAS or CICSPlex.

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

**FROM(data-area)**

Identifies a buffer containing a resource table record that represents the definition to be updated.

The record must include all of the attributes for the resource table specified on the OBJECT option. For optional attributes that you do not want to specify, set the field to null (that is, zero) values.

**LENGTH(data-value)**

A fullword value that specifies the length of the FROM or MODIFY buffer.

Note: The buffer length you specify should not include any data other than a resource table record or modification expression.

**MODIFY(data-area)**

Identifies a buffer containing the modification expression to be used in modifying CICS Definition resource table records.

For details on how to form a modification expression, see CICSPlex 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 \textit{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 \textit{CICSPlex System Manager Application Programming Guide}. For a description of the parameters that are valid for a given resource table, see the \textit{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 \textit{cpsm-token} value that identifies a thread is returned by the CONNECT command.

Conditions

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

OK
The command completed processing successfully.

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

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

NOSTORAGE
The application stub program could not obtain the necessary storage in the address space where the processing thread is running.
REQTIMEOUT
One of the CMASs or MASs to which the request was directed did not respond.

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.

INVALIDDATA
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.

INVALIDCMD
The command is invalid for the following reason:

LENGTH
The total length of all the options on the command exceeds the maximum limit.

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

CONTEXT
FROM
LENGTH
MODIFY
OBJECT
PARM
PARMLEN
RESULT
THREAD.

Check the command description for valid parameter syntax.

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

APITASK
The API control subtask is not active.

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

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

MAINTPOINT
The maintenance point for the current context is not available.

NOTPERMIT
A not permitted condition occurred for the following reason:

USRID
The user ID associated with the processing thread does not have the required security authorization.
SERVERGONE
   The CMAS to which the processing thread was connected is no longer active.

TABLEERROR
   A resource table record is invalid for one of the following reasons:
   DATAERROR
       The value associated with one or more resource table attributes is invalid. This error can occur if:
       - The resource table is missing required attributes, contains one or more conflicting attributes, or does not exist.
       - A CICS resource definition contains attributes that would cause the EXEC CICS CREATE command to issue warnings.

       Use the FEEDBACK command to retrieve additional data about this error.
   INVALIDATTR
       One of the resource table attributes is invalid.
   INVALIDVER
       The specified version of the resource table is not supported by CICSPlex SM.

VERSIONINVL
   A version conflict occurred for one of the following reasons:
   NOTSUPPORTED
       The version of the application stub program used for this command is not supported.
   NOTVSNCNON
       The version of the application stub program used for this command is not the same as the version used with the CONNECT command.
Chapter 3. REXX functions and commands

This chapter contains detailed descriptions of the REXX functions and commands supplied with CICSPlex SM. These functions and commands can be used only with the REXX run-time interface.

The REXX functions supplied with CICSPlex SM make use of standard REXX variable substitution rules. In addition to REXX return codes, these functions can produce EYUARnnnn messages. For descriptions of those messages, see CICSPlex 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
EYUAPI()

**EYUAPI()**

Passes an API command to CICSPlex SM.

```plaintext
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.

\texttt{var = EYUINIT()}

**Description**

This command initializes the CICSPlex SM API environment and allocates the necessary resources. EYUINIT should be the first function issued in a REXX program.

**Note:** You must issue an EYUINIT or EYUAPI function before you can use the ADDRESS CPSM command to pass API commands to REXX.

**Return codes**

The following is a list of the REXX return codes that can be returned by the EYUINIT function in its assigned variable (var).

- **0** The EYUINIT function was successful.
- **1** The EYUINIT function failed.
EYUREAS()

Translate the numeric value returned by the REASON option of an API command.

\texttt{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 (\texttt{var}).

\texttt{nnnn}  The numeric or character equivalent of the specified REASON value.
\texttt{−1}  The specified REASON value is invalid and could not be translated.
EYURESP()

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).

*n invit* The numeric or character equivalent of the specified RESPONSE value.

−1 The specified RESPONSE value is invalid and could not be translated.
EYUTERM()

Description
This command terminates the CICSPlex SM API environment and releases any allocated resources. EYUTERM should be the last function issued in a REXX program.

Note: If the CICSPlex SM host subcommand environment is actually installed at your enterprise (as opposed to being called from the function package), you may not need to use EYUTERM at the end of every program. The resources that remain allocated can be reused by the next program that accesses the host subcommand environment.

Return codes
The following is a list of the REXX return codes that can be returned by the EYUTERM function in its assigned variable (var).

0  The EYUTERM function was successful.
1  The EYUTERM function failed.
Commands

The REXX-specific commands supplied with CICSPlex SM perform a series of API commands internally and return the results to REXX.

The commands are presented here in alphabetical order. Each description includes the purpose, syntax, and available options for the command.

Notes:
1. You cannot use these commands to process user-defined views of a resource table that were created by the SPECIFY VIEW command. If you create a view with the same name as a supplied resource table and then specify that name on one of these commands, the command fails.
2. These commands do not use the RESPONSE and REASON options. The result of these REXX-specific processes is returned by the STATUS option.
3. These commands do not provide any useful FEEDBACK information. The API commands that are issued internally reuse the same feedback area. So, when one of these commands finishes processing, the feedback area does not represent the entire sequence of events.

The commands are:
- "TBUILD" on page 136
- "TPARSE" on page 138
Build a resource table record from a set of variables.

```plaintext
TBUILD—OBJECT(data-value)—PREFIX(data-value)—STATUS(data-ref)
VAR(data-area)—ASIS—THREAD(cpsm-token)
```

**Description**

This command builds a resource table record from a set of variables that represent the individual attributes of a CICSPlex SM or CICS definition. A definition is represented by a resource table with a type of CPSM Definition or CICS Definition.

You form the attribute variables by adding a prefix to the attribute name, like this:

```plaintext
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.

**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

TPARSE

Parse a resource table record from a variable into a set of variables.

```
TPARSE—OBJECT(data-value)—PREFIX(data-value)—STATUS(data-ref)

VAR(data-area)     ASIS

THREAD(cpsm-token)
```

Description

This command parses a resource table record from a variable into a set of variables that represent the individual attributes of the table. You can use TPARSE with any type of CICSPlex SM resource table.

The resource table variable can be any valid REXX variable, including a stem variable. The output variables are formed by adding a prefix to the attribute name, like this:

```
prefix_attribute
```

where prefix is a text string that you supply and attribute is the name of an attribute in the resource table. An underscore (_) is inserted between the prefix and the attribute name.

Note: For complete descriptions of the resource tables and their attributes, see the CICSPlex 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.
TPARSE
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 149. 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>CANCEL</td>
<td>OK</td>
<td>ENVIRONERROR NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
</tr>
<tr>
<td>CONNECT</td>
<td>OK</td>
<td>ENVIRONERROR APITASKERR, NOSERVICE, NOSTORAGE, SOCRESOURCE, SOCRESOURCE, SOLRESOURCE</td>
</tr>
<tr>
<td>COPY</td>
<td>OK</td>
<td>NODATA</td>
</tr>
<tr>
<td>CREATE</td>
<td>OK</td>
<td>ENVIRONERROR NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE</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>DELETE</td>
<td>OK</td>
<td>NODATA</td>
</tr>
<tr>
<td></td>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td></td>
<td>ENVIRONEROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCES, SOLRESOURCES</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INUSE</td>
<td>FILTER, VIEW</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>FILTER, RESULT, THREAD, VIEW</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>VERSIONINVLL</td>
<td>NOSTARTED, NOTVSNCONN</td>
</tr>
<tr>
<td>DISCARD</td>
<td>OK</td>
<td>NODATA</td>
</tr>
<tr>
<td></td>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td></td>
<td>ENVIRONEROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCES</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INUSE</td>
<td>FILTER, VIEW</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>FILTER, RESULT, THREAD, VIEW</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>VERSIONINVLL</td>
<td>NOSTARTED, NOTVSNCONN</td>
</tr>
<tr>
<td>DISCONNECT</td>
<td>OK</td>
<td>NODATA</td>
</tr>
<tr>
<td></td>
<td>ENVIRONMENT</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCES</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>THREAD</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>VERSIONINVLL</td>
<td>NOSTARTED, NOTVSNCONN</td>
</tr>
<tr>
<td>FEEDBACK</td>
<td>OK</td>
<td>NODATA</td>
</tr>
<tr>
<td></td>
<td>WARNING</td>
<td>AREATOOSMALL</td>
</tr>
<tr>
<td></td>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td></td>
<td>ENVIRONMENT</td>
<td>NOSERVICE, NOSTORAGE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>COUNT, INTO, LENGTH, RESULT, THREAD</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>VERSIONINVLL</td>
<td>NOSTARTED, NOTVSNCONN</td>
</tr>
<tr>
<td>FETCH</td>
<td>OK</td>
<td>NODATA</td>
</tr>
<tr>
<td></td>
<td>WARNING</td>
<td>AREATOOSMALL</td>
</tr>
<tr>
<td></td>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td></td>
<td>ENVIRONMENT</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCES, SOLRESOURCES</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>COUNT, FILTER, INTO, LENGTH, NOTFILTER, POSITION, RESULT, THREAD</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</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>GET</td>
<td>OK</td>
<td>BUSY, STAT, ENVIRONERROR, FAILED, INVALIDDATA, INVALIDPARM, NOTAVAIL, NOTFOUND, NOTPERMIT, TABLEERROR, VERSIONINVNL, SERVERGONE</td>
</tr>
<tr>
<td></td>
<td>SCHEDULED</td>
<td>NODATA</td>
</tr>
<tr>
<td></td>
<td>NODATA</td>
<td>RESULT</td>
</tr>
<tr>
<td></td>
<td>ERROR</td>
<td>DATAERROR, INVALIDVER</td>
</tr>
<tr>
<td></td>
<td>GETDEF</td>
<td>NODATA</td>
</tr>
<tr>
<td></td>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td></td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDDATA</td>
<td>CRITERIA</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>FILTER, LENGTH, CONTEXT, CRITERIA, FILTER, LENGTH, OBJECT, PARM, PARMLEN, RESULT, SCOPE, THREAD, TOKEN</td>
</tr>
<tr>
<td></td>
<td>NOTAVAIL</td>
<td>APITASK, CMAS, CPSMAPI, MAINTPOINT, SCOPE, WORKLOAD</td>
</tr>
<tr>
<td></td>
<td>NOTFOUND</td>
<td>ATTRIBUTE</td>
</tr>
<tr>
<td></td>
<td>NOTPERMIT</td>
<td>USRID</td>
</tr>
<tr>
<td></td>
<td>TABLEERROR</td>
<td>DATAERROR, INVALIDVER</td>
</tr>
<tr>
<td></td>
<td>VERSIONINVNL</td>
<td>NOTSUPPORTED, NOTVSNConn</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td>NOTSUPPORTED, NOTVSNConn</td>
</tr>
<tr>
<td></td>
<td>TABLEERROR</td>
<td>DATAERROR, INVALIDVER</td>
</tr>
<tr>
<td>GROUP</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, SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDCMD</td>
<td>LENGTH</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>CONTEXT, EVENT, FILTER, NOTFILTER, NOTIFICATION, THREAD, TO</td>
</tr>
<tr>
<td></td>
<td>NOTAVAIL</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td>TABLEERROR</td>
<td>INVALIDVER</td>
</tr>
<tr>
<td></td>
<td>VERSIONINVNL</td>
<td>NOTSUPPORTED, NOTVSNConn</td>
</tr>
<tr>
<td>LISTEN</td>
<td>OK</td>
<td>NODATA</td>
</tr>
<tr>
<td></td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INCOMPATIBLE</td>
<td>INVALIDEV</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>CONTEXT, EVENT, FILTER, NOTFILTER, NOTIFICATION, THREAD, TOKEN</td>
</tr>
<tr>
<td></td>
<td>NOTAVAIL</td>
<td>APITASK, CPSMAPI, PLEXMGR</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td>NOTSUPPORTED, NOTVSNConn</td>
</tr>
<tr>
<td></td>
<td>VERSIONINVNL</td>
<td>NOTSUPPORTED, NOTVSNConn</td>
</tr>
<tr>
<td>LOCATE</td>
<td>OK</td>
<td>NODATA</td>
</tr>
</tbody>
</table>

Appendix A. RESPONSE and REASON values 143
### RESPONSE and REASON values

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>RESPONSE</th>
<th>REASONS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NODATA</td>
<td>BACKWARD, FORWARD</td>
</tr>
<tr>
<td></td>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td></td>
<td>ENVIRONER</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE, SOLRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>BACKWARD, FILTER, FORWARD, NOTFILTER, POSITION, 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, NOTVSNCNN</td>
</tr>
<tr>
<td>MARK</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>ENVIRONER</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE, SOLRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, 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, NOTVSNCNN</td>
</tr>
<tr>
<td>ORDER</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td></td>
<td>ENVIRONER</td>
<td>NOSERVICE, NOSTORAGE SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDCMD</td>
<td>LENGTH</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>BY, LENGTH, 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, NOTVSNCNN</td>
</tr>
<tr>
<td>PERFORM OBJECT</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCHEDULED</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NODATA</td>
<td>BACKWARD, FORWARD</td>
</tr>
<tr>
<td></td>
<td>BUSY</td>
<td>RESULT</td>
</tr>
<tr>
<td></td>
<td>ENVIRONER</td>
<td>NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDDATA</td>
<td>PARM, CRITERIA</td>
</tr>
<tr>
<td></td>
<td>INVALIDCMD</td>
<td>FILTER,LENGTH</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>ACTION, CONTEXT, CRITERIA, FILTER, LENGTH, OBJECT, PARM, PARMLEN, RESULT, SCOPE, THREAD, TOKEN</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CMAS, CPSMAPI, MAINTPOINT, PLEXMGR, SCOPE, WORKLOAD</td>
</tr>
<tr>
<td></td>
<td>NOTFOUND</td>
<td>ACTION, ATTRIBUTE</td>
</tr>
<tr>
<td></td>
<td>NOTPERMIT</td>
<td>USRID</td>
</tr>
<tr>
<td></td>
<td>SERVERGONE</td>
<td>DATAERROR</td>
</tr>
<tr>
<td></td>
<td>TABLEERROR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VERSIONINVL</td>
<td>ACTION, NOTSUPPORTED, NOTVSNCNN</td>
</tr>
<tr>
<td></td>
<td>WARNING</td>
<td>RESULT, ACTION</td>
</tr>
<tr>
<td>PERFORM SET</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCHEDULED</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NODATA</td>
<td>BACKWARD, FORWARD</td>
</tr>
<tr>
<td></td>
<td>BUSY</td>
<td>RESULT</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></td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, REQTIMEOUT SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td>INVALIDDATA</td>
<td>PARM, CRITERIA</td>
</tr>
<tr>
<td></td>
<td>INVALIDCMD</td>
<td>LENGTH</td>
</tr>
<tr>
<td></td>
<td>INVALIDPARM</td>
<td>ACTION, FILTER, NOTFILTER, PARM, PARMLEN, POSITION, RESULT, THREAD, TOKEN</td>
</tr>
<tr>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CMAS, CPSMAPI, MAINTPOINT, PLEXMGR, SCOPE, WORKLOAD</td>
</tr>
<tr>
<td></td>
<td>NOTFOUND</td>
<td>ACTION, ATTRIBUTE</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</td>
</tr>
<tr>
<td></td>
<td>VERSIONINVL</td>
<td>ACTION, NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

#### QUALIFY

<table>
<thead>
<tr>
<th></th>
<th>OK</th>
<th>ENVIROERROR</th>
<th>NOSERVICE, NOSTORAGE, SOCRESOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INVALIDPARM</td>
<td>CONTEXT, SCOPE, THREAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI, PLEXMGR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

#### QUERY

<table>
<thead>
<tr>
<th></th>
<th>OK</th>
<th>BUSY</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INVALIDPARM</td>
<td>CONTEXT, DATALENGTH, OBJECT, RESULT, THREAD, TYPE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

#### RECEIVE

<table>
<thead>
<tr>
<th></th>
<th>OK</th>
<th>NODATA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>WARNING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENVIRONERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FAILED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INVALIDPARM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTAVAILABLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SERVERGONE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VERSIONINVL</td>
</tr>
</tbody>
</table>

#### REFRESH

<table>
<thead>
<tr>
<th></th>
<th>OK</th>
<th>SCHEDULED</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BUSY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INVALIDPARM</td>
<td>COUNT, FILTER, NOTFILTER, RESULT, THREAD, TOKEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTAVAILABLE</td>
<td>APITASK, CMAS, CPSMAPI, MAINTPOINT, SCOPE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTPERMIT</td>
<td>USRID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SERVERGONE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TABLEERROR</td>
<td>DATAERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
</tr>
</tbody>
</table>

#### REMOVE

|          | OK | ENVIRONERROR | NOSERVICE, NOSTORAGE, REQTIMEOUT, SOCRESOURCE |

---

Appendix A. RESPONSE and REASON values 145
### RESPONSE and REASON values

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>RESPONSE</th>
<th>REASONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
<td></td>
</tr>
<tr>
<td>INVALIDCMD</td>
<td>LENGTH</td>
<td></td>
</tr>
<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></td>
<td></td>
</tr>
<tr>
<td>TABLEERROR</td>
<td>DATAERROR, INVALIDATTR, INVALIDVER</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
<td></td>
</tr>
</tbody>
</table>

#### SET

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

#### SPECIFY FILTER

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
<td></td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
<td></td>
</tr>
<tr>
<td>INVALIDCMD</td>
<td>FILTER,LENGTH</td>
<td></td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>CRITERIA, FILTER, LENGTH, OBJECT, THREAD</td>
<td></td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
<td></td>
</tr>
<tr>
<td>NOTFOUND</td>
<td>ATTRIBUTE</td>
<td></td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
<td></td>
</tr>
</tbody>
</table>

#### SPECIFY VIEW

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DUPE</td>
<td>VIEW</td>
<td></td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
<td></td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
<td></td>
</tr>
<tr>
<td>INVALIDDATA</td>
<td>CRITERIA</td>
<td></td>
</tr>
<tr>
<td>INVALIDCMD</td>
<td>LENGTH</td>
<td></td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>ATTRIBUTE, FIELDS, LENGTH, OBJECT, THREAD, VIEW</td>
<td></td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CPSMAPI</td>
<td></td>
</tr>
<tr>
<td>NOTFOUND</td>
<td>ATTRIBUTE</td>
<td></td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLEERROR</td>
<td>DATAERROR, INVALIDVER</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
<td></td>
</tr>
</tbody>
</table>

#### TERMINATE

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE</td>
<td></td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
<td></td>
</tr>
</tbody>
</table>

#### TRANSLATE

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURCE</td>
<td></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>FAILED</td>
<td>ABENDED, EXCEPTION</td>
<td></td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>ATTRIBUTE, FROMCHAR, FROMCV, OBJECT, THREAD, TOCHAR, TOCV</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>TABLEERROR</td>
<td>INVALIDVER, INVALIDVER</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
<td></td>
</tr>
</tbody>
</table>

**UNMARK**

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>RESPONSE</th>
<th>REASONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NODATA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WARNING</td>
<td>AREATOOSMALL, DATAERROR</td>
<td></td>
</tr>
<tr>
<td>BUSY</td>
<td>RESULT</td>
<td></td>
</tr>
<tr>
<td>ENVIRONERROR</td>
<td>NOSERVICE, NOSTORAGE, SOCRESOURSE, SOLRESOURSE</td>
<td></td>
</tr>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
<td></td>
</tr>
<tr>
<td>INVALIDCMD</td>
<td>LENGTH</td>
<td></td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>COUNT, FILTER, INTO, LENGTH, NOTFILTER, PARM, PARMLEN, 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>
</tbody>
</table>

**UPDATE**

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>RESPONSE</th>
<th>REASONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED</td>
<td>ABENDED, EXCEPTION</td>
<td></td>
</tr>
<tr>
<td>INVALIDCMD</td>
<td>LENGTH</td>
<td></td>
</tr>
<tr>
<td>INVALIDPARM</td>
<td>CONTEXT, FROM, LENGTH, MODIFY, OBJECT, PARM, PARMLEN, RESULT, THREAD</td>
<td></td>
</tr>
<tr>
<td>NOTAVAILABLE</td>
<td>APITASK, CMAS, CPSMAPI</td>
<td></td>
</tr>
<tr>
<td>NOTPERMIT</td>
<td>USRID</td>
<td></td>
</tr>
<tr>
<td>SERVERGONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLEERROR</td>
<td>DATAERROR, INVALIDATTR, INVALIDVER</td>
<td></td>
</tr>
<tr>
<td>VERSIONINVL</td>
<td>NOTSUPPORTED, NOTVSNCONN</td>
<td></td>
</tr>
</tbody>
</table>
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 1–474. 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 180.

REASON
Values returned by the REASON option of an API command. These EYUDAs have numeric values in the range of 1280–1370. See “EYUDA REASON values in numerical order” on page 181.

EYUDA information is provided in the following sections:
- “EYUDA general values in numerical order”
- “EYUDA general values in alphabetic order” on page 162
- “EYUDA RESPONSE values in numerical order” on page 180
- “EYUDA RESPONSE values in alphabetic order” on page 180
- “EYUDA REASON values in numerical order” on page 181
- “EYUDA REASON values in alphabetic order” on page 183

Note: The EYUDA values and their numeric equivalents listed for HOTPOOL are valid only in CICS Transaction Server 2.2 and 2.3.

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>
</tbody>
</table>
### Numeric EYUDA general values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>LW</td>
</tr>
<tr>
<td>12</td>
<td>NM</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>30</td>
<td>SUSPEND</td>
</tr>
<tr>
<td>33</td>
<td>VALUE</td>
</tr>
<tr>
<td>34</td>
<td>THRESHOLD</td>
</tr>
<tr>
<td>35</td>
<td>SAM</td>
</tr>
<tr>
<td>36</td>
<td>APM</td>
</tr>
<tr>
<td>37</td>
<td>MRM</td>
</tr>
<tr>
<td>38</td>
<td>TRANID</td>
</tr>
<tr>
<td>39</td>
<td>TERMID</td>
</tr>
<tr>
<td>40</td>
<td>SIGNID</td>
</tr>
<tr>
<td>41</td>
<td>RACFGID</td>
</tr>
<tr>
<td>42</td>
<td>USERID</td>
</tr>
<tr>
<td>43</td>
<td>NULL</td>
</tr>
<tr>
<td>44</td>
<td>CHARSTR</td>
</tr>
<tr>
<td>45</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>46</td>
<td>INACTIVE</td>
</tr>
<tr>
<td>47</td>
<td>WAITING</td>
</tr>
<tr>
<td>48</td>
<td>QUIESCING</td>
</tr>
<tr>
<td>49</td>
<td>POOL</td>
</tr>
<tr>
<td>50</td>
<td>LTRAN</td>
</tr>
</tbody>
</table>
### Numeric EYUDA general values

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>RTRAN</td>
</tr>
<tr>
<td>53</td>
<td>GOAL</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>MJRNL</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>
</tbody>
</table>
## Numeric EYUDA general values

<table>
<thead>
<tr>
<th><code>92</code></th>
<th>INDIRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>93</code></td>
<td>MRO</td>
</tr>
<tr>
<td><code>94</code></td>
<td>NOTAPPLIC</td>
</tr>
<tr>
<td><code>95</code></td>
<td>LFILE</td>
</tr>
<tr>
<td><code>96</code></td>
<td>RFILE</td>
</tr>
<tr>
<td><code>97</code></td>
<td>CTABL</td>
</tr>
<tr>
<td><code>98</code></td>
<td>UTABL</td>
</tr>
<tr>
<td><code>99</code></td>
<td>INSTALLED</td>
</tr>
<tr>
<td><code>100</code></td>
<td>PENDING</td>
</tr>
<tr>
<td><code>101</code></td>
<td>INHERIT</td>
</tr>
<tr>
<td><code>102</code></td>
<td>EXPLICIT</td>
</tr>
<tr>
<td><code>103</code></td>
<td>CICSSYS</td>
</tr>
<tr>
<td><code>104</code></td>
<td>SYSGROUP</td>
</tr>
<tr>
<td><code>105</code></td>
<td>KEEP</td>
</tr>
<tr>
<td><code>106</code></td>
<td>NAME</td>
</tr>
<tr>
<td><code>107</code></td>
<td>FORCE</td>
</tr>
<tr>
<td><code>108</code></td>
<td>NONE</td>
</tr>
<tr>
<td><code>109</code></td>
<td>UNASSIGNED</td>
</tr>
<tr>
<td><code>110</code></td>
<td>DROP</td>
</tr>
<tr>
<td><code>111</code></td>
<td>LOCAL</td>
</tr>
<tr>
<td><code>112</code></td>
<td>REMOTE</td>
</tr>
<tr>
<td><code>113</code></td>
<td>DEFAULT</td>
</tr>
<tr>
<td><code>114</code></td>
<td>REMOVE</td>
</tr>
<tr>
<td><code>115</code></td>
<td>DORMANT</td>
</tr>
<tr>
<td><code>116</code></td>
<td>START</td>
</tr>
<tr>
<td><code>117</code></td>
<td>END</td>
</tr>
<tr>
<td><code>118</code></td>
<td>ADJACENT</td>
</tr>
<tr>
<td><code>119</code></td>
<td>LOSTCON</td>
</tr>
<tr>
<td><code>120</code></td>
<td>CREATING</td>
</tr>
<tr>
<td><code>121</code></td>
<td>REMOVING</td>
</tr>
<tr>
<td><code>122</code></td>
<td>QUIESCED</td>
</tr>
<tr>
<td><code>123</code></td>
<td>LINKACTIVE</td>
</tr>
<tr>
<td><code>124</code></td>
<td>LINKDOWN</td>
</tr>
<tr>
<td><code>125</code></td>
<td>ESSS</td>
</tr>
<tr>
<td><code>126</code></td>
<td>CONACT</td>
</tr>
<tr>
<td><code>127</code></td>
<td>RESET</td>
</tr>
<tr>
<td><code>128</code></td>
<td>SYSDUMP</td>
</tr>
</tbody>
</table>
Numeric EYUDA general values

129  TRANDUMP
130  MAXTASK
131  STALLED
132  SOSUDSA
133  SOSCDSA
134  SOSEUDSA
135  SOSECDSA
136  SOSERDSA
137  SOSSDSA
138  SOSESDSA
139  SORRDSA
140  QUIESCE
141  PRIMARY
142  SECONDARY
143  DUPLICATE
144  FROZEN
145  ALL
146  ANY
147  SUM
148  MIN
149  MAX
150  AVG
151  CNT
152  LE
153  GE
154  SDSA
155  ESDSA
156  RDSA
157  SOSMVS
158  SOSBELOW
159  SOSABOVE
160  EXECUTE
161  CHECK
162  LOSTCMAS
163  LOSTMAS
164  AASTERISK
165  BLANK
### Numeric EYUDA general values

<table>
<thead>
<tr>
<th>Page</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td>172</td>
<td>AFTER</td>
</tr>
<tr>
<td>173</td>
<td>ALLREQS</td>
</tr>
<tr>
<td>174</td>
<td>ASA</td>
</tr>
<tr>
<td>177</td>
<td>ASIS</td>
</tr>
<tr>
<td>178</td>
<td>ASSEMBLER</td>
</tr>
<tr>
<td>179</td>
<td>BACKOUTONLY</td>
</tr>
<tr>
<td>180</td>
<td>BEFORE</td>
</tr>
<tr>
<td>181</td>
<td>BLUE</td>
</tr>
<tr>
<td>182</td>
<td>C</td>
</tr>
<tr>
<td>183</td>
<td>CLEARCONV</td>
</tr>
<tr>
<td>184</td>
<td>CLOSE</td>
</tr>
<tr>
<td>185</td>
<td>COBOL</td>
</tr>
<tr>
<td>186</td>
<td>COLD</td>
</tr>
<tr>
<td>187</td>
<td>CYCLIC</td>
</tr>
<tr>
<td>188</td>
<td>DEFERRED</td>
</tr>
<tr>
<td>189</td>
<td>DIP</td>
</tr>
<tr>
<td>190</td>
<td>DISK</td>
</tr>
<tr>
<td>191</td>
<td>EODS</td>
</tr>
<tr>
<td>192</td>
<td>EXTA</td>
</tr>
<tr>
<td>193</td>
<td>FILE</td>
</tr>
<tr>
<td>194</td>
<td>FIRSTREF</td>
</tr>
<tr>
<td>195</td>
<td>GREEN</td>
</tr>
<tr>
<td>196</td>
<td>IDENTIFY</td>
</tr>
<tr>
<td>197</td>
<td>IGNORE</td>
</tr>
<tr>
<td>198</td>
<td>INITIAL</td>
</tr>
<tr>
<td>199</td>
<td>INOUT</td>
</tr>
<tr>
<td>200</td>
<td>INPUT</td>
</tr>
<tr>
<td>201</td>
<td>LEAVE</td>
</tr>
<tr>
<td>202</td>
<td>LE370</td>
</tr>
<tr>
<td>203</td>
<td>LINEAR</td>
</tr>
<tr>
<td>204</td>
<td>LINK</td>
</tr>
<tr>
<td>205</td>
<td>LMS</td>
</tr>
</tbody>
</table>
## Numeric EYUDA general values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGICAL</td>
<td>Logical value</td>
</tr>
<tr>
<td>LOGOFF</td>
<td>Logoff value</td>
</tr>
<tr>
<td>LRU</td>
<td>Least Recently Used value</td>
</tr>
<tr>
<td>MACHINE</td>
<td>Machine value</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>Message value</td>
</tr>
<tr>
<td>MIXIDPE</td>
<td>Mixidpe value</td>
</tr>
<tr>
<td>MOD</td>
<td>Mod value</td>
</tr>
<tr>
<td>MODIFYREQS</td>
<td>Modifyreqs value</td>
</tr>
<tr>
<td>MSRE</td>
<td>Msre value</td>
</tr>
<tr>
<td>NEUTRAL</td>
<td>Neutral value</td>
</tr>
<tr>
<td>NEW</td>
<td>New value</td>
</tr>
<tr>
<td>NOFORCE</td>
<td>Nonforce value</td>
</tr>
<tr>
<td>NONVTAM</td>
<td>Nonvtam value</td>
</tr>
<tr>
<td>OLD</td>
<td>Old value</td>
</tr>
<tr>
<td>ONLY</td>
<td>Only value</td>
</tr>
<tr>
<td>OPEN</td>
<td>Open value</td>
</tr>
<tr>
<td>OPID</td>
<td>Opid value</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>Output value</td>
</tr>
<tr>
<td>PERSISTENT</td>
<td>Persistent value</td>
</tr>
<tr>
<td>PHYSICAL</td>
<td>Physical value</td>
</tr>
<tr>
<td>PINK</td>
<td>Pink value</td>
</tr>
<tr>
<td>PLI</td>
<td>Pli value</td>
</tr>
<tr>
<td>PRINTER</td>
<td>Printer value</td>
</tr>
<tr>
<td>RED</td>
<td>Red value</td>
</tr>
<tr>
<td>RECOVERY</td>
<td>Recovery value</td>
</tr>
<tr>
<td>REJECT</td>
<td>Reject value</td>
</tr>
<tr>
<td>RELEASESESS</td>
<td>Release sess value</td>
</tr>
<tr>
<td>REREAD</td>
<td>Reread value</td>
</tr>
<tr>
<td>RPG</td>
<td>Rpg value</td>
</tr>
<tr>
<td>SCS</td>
<td>Scs value</td>
</tr>
<tr>
<td>SECURITY</td>
<td>Security value</td>
</tr>
<tr>
<td>SHR</td>
<td>Shr value</td>
</tr>
<tr>
<td>SKIP</td>
<td>Skip value</td>
</tr>
<tr>
<td>SPECIFIC</td>
<td>Specific value</td>
</tr>
<tr>
<td>STARTIO</td>
<td>Startio value</td>
</tr>
<tr>
<td>STARTUP</td>
<td>Startup value</td>
</tr>
<tr>
<td>STRFIELD</td>
<td>Strfield value</td>
</tr>
</tbody>
</table>
Numeric EYUDA general values

243  SYSDEFAULT
244  TAPE
245  TERMINAL
246  TRANSACTION
247  TRANSIENT
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  JRNDEF
279  LSRDEF
Numeric EYUDA general values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>280</td>
<td>MAPDEF</td>
</tr>
<tr>
<td>281</td>
<td>PARTDEF</td>
</tr>
<tr>
<td>282</td>
<td>PRTNDEF</td>
</tr>
<tr>
<td>283</td>
<td>PROFDEF</td>
</tr>
<tr>
<td>284</td>
<td>PROGDEF</td>
</tr>
<tr>
<td>285</td>
<td>SESSDEF</td>
</tr>
<tr>
<td>286</td>
<td>TDQDEF</td>
</tr>
<tr>
<td>287</td>
<td>TERMDEF</td>
</tr>
<tr>
<td>288</td>
<td>TRANDEF</td>
</tr>
<tr>
<td>289</td>
<td>TRNCLDEF</td>
</tr>
<tr>
<td>290</td>
<td>TSQDEF</td>
</tr>
<tr>
<td>291</td>
<td>TYPTMDEF</td>
</tr>
<tr>
<td>292</td>
<td>MPSYNCCR</td>
</tr>
<tr>
<td>293</td>
<td>ASSOCIATIONS</td>
</tr>
<tr>
<td>294</td>
<td>MEMBERS</td>
</tr>
<tr>
<td>295</td>
<td>DB2CDEF</td>
</tr>
<tr>
<td>296</td>
<td>DB2EDEF</td>
</tr>
<tr>
<td>297</td>
<td>DB2TDEF</td>
</tr>
<tr>
<td>298</td>
<td>FSEGDEF</td>
</tr>
<tr>
<td>299</td>
<td>TSMDEF</td>
</tr>
<tr>
<td>300</td>
<td>ENQMDEF</td>
</tr>
<tr>
<td>301</td>
<td>TCPDEF</td>
</tr>
<tr>
<td>302</td>
<td>DOCDEF</td>
</tr>
<tr>
<td>303</td>
<td>FULL</td>
</tr>
<tr>
<td>304</td>
<td>RELEASE</td>
</tr>
<tr>
<td>305</td>
<td>PA1</td>
</tr>
<tr>
<td>306</td>
<td>PA2</td>
</tr>
<tr>
<td>307</td>
<td>PA3</td>
</tr>
<tr>
<td>308</td>
<td>PF1</td>
</tr>
<tr>
<td>309</td>
<td>PF2</td>
</tr>
<tr>
<td>310</td>
<td>PF3</td>
</tr>
<tr>
<td>311</td>
<td>PF4</td>
</tr>
<tr>
<td>312</td>
<td>PF5</td>
</tr>
<tr>
<td>313</td>
<td>PF6</td>
</tr>
<tr>
<td>314</td>
<td>PF7</td>
</tr>
<tr>
<td>315</td>
<td>PF8</td>
</tr>
<tr>
<td>316</td>
<td>PF9</td>
</tr>
</tbody>
</table>
Numeric EYUDA general values

317  PF10
318  PF11
319  PF12
320  PF13
321  PF14
322  PF15
323  PF16
324  PF17
325  PF18
326  PF19
327  PF20
328  PF21
329  PF22
330  PF23
331  PF24
332  STANDARD
333  APPC
334  BATCHDI
335  BCHLU
336  CONSOLE
337  CONTLU
338  INTLU
339  LUTYPE2
340  LUTYPE3
341  LUTYPE4
342  L3277
343  L3284
344  L3286
345  PIPELINE
346  SCSPRINT
347  TLX
348  TWX
349  USERPROG
350  3270P
351  3275
352  3277
353  3277CM
Numeric EYUDA general values

354  3284
355  3284CM
356  3286
357  3286CM
358  3600
359  3614
360  3650
361  3653
362  3767
363  3767C
364  3767I
365  3770
366  3770B
367  3770C
368  3770I
369  3790
370  BLINK
371  REVERSE
372  UNDERLINE
373  INSTALL
374  RESDEF
375  RASINDSC
376  RESTYPE
377  SCOPETYP
378  REBUILD
379  RECONNECT
380  CONNECTING
381  TWAIT
382  NOTWAIT
383  DISCONNING
384  CFTBL
385  CF
386  QUASIRENT
387  THREADSAFE
388  PROCDEF
389  BAPPL
390  ACTIVITY
Numeric EYUDA general values

391  PROCESS
393  PMWINDOW
394  FULLSCREEN
395  COM1
396  COM2
397  COM3
398  COM4
399  COM5
400  COM6
401  COM7
402  COM8
403  CLIENTAUTH
404  RQMDEF
405  3270TERM
406  3270PRNT
407  3270DBTM
408  3270DBPR
409  3151TERM
410  SEQTERM
411  EOF
412  EOT
413  FEPOODEF
414  FETRGDEF
415  FENODDEF
416  FEPRODEF
417  DEBUG
418  LEVSE
419  KEY
420  RBA
423  EJCODEF
424  EJDJDEF
425  BASIC
426  CERTIFICATE
427  AUTOREGISTER
428  AUTOMATIC
429  CLIENTCERT
430  LINK3270
Numeric EYUDA general values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>431</td>
<td>FACILITY</td>
</tr>
<tr>
<td>432</td>
<td>NOTOPEN</td>
</tr>
<tr>
<td>433</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>434</td>
<td>REENTPROT</td>
</tr>
<tr>
<td>435</td>
<td>NOREENTPROT</td>
</tr>
<tr>
<td>436</td>
<td>ASSERTED</td>
</tr>
<tr>
<td>437</td>
<td>MIRROR</td>
</tr>
<tr>
<td>438</td>
<td>DPL</td>
</tr>
<tr>
<td>439</td>
<td>ONCRPC</td>
</tr>
<tr>
<td>440</td>
<td>WEB</td>
</tr>
<tr>
<td>441</td>
<td>BRIDGE</td>
</tr>
<tr>
<td>442</td>
<td>CICSBTS</td>
</tr>
<tr>
<td>443</td>
<td>TDQUEUE</td>
</tr>
<tr>
<td>444</td>
<td>TERMSTART</td>
</tr>
<tr>
<td>445</td>
<td>XMRUN</td>
</tr>
<tr>
<td>446</td>
<td>SOCKET</td>
</tr>
<tr>
<td>447</td>
<td>RRS</td>
</tr>
<tr>
<td>448</td>
<td>IIRQRECVR</td>
</tr>
<tr>
<td>449</td>
<td>RZSTTRPT</td>
</tr>
<tr>
<td>450</td>
<td>IIOP</td>
</tr>
<tr>
<td>451</td>
<td>ECI</td>
</tr>
<tr>
<td>452</td>
<td>HOTPOOL</td>
</tr>
<tr>
<td>453</td>
<td>JVM</td>
</tr>
<tr>
<td>454</td>
<td>HTASK</td>
</tr>
<tr>
<td>455</td>
<td>IOERROR</td>
</tr>
<tr>
<td>456</td>
<td>INVALIDFILE</td>
</tr>
<tr>
<td>457</td>
<td>SUSPENDED</td>
</tr>
<tr>
<td>458</td>
<td>SUSPENDING</td>
</tr>
<tr>
<td>459</td>
<td>HISTORY</td>
</tr>
<tr>
<td>460</td>
<td>REALTIME</td>
</tr>
<tr>
<td>461</td>
<td>VELOCITY</td>
</tr>
<tr>
<td>462</td>
<td>DISCRETIONRY</td>
</tr>
<tr>
<td>463</td>
<td>CONNECTED</td>
</tr>
<tr>
<td>464</td>
<td>NOTCONNECTED</td>
</tr>
<tr>
<td>465</td>
<td>RESUMING</td>
</tr>
<tr>
<td>466</td>
<td>URIMPDEF</td>
</tr>
<tr>
<td>467</td>
<td>PIPEDEF</td>
</tr>
</tbody>
</table>
**Numeric EYUDA general values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>468</td>
<td>WEBSVDEF</td>
</tr>
<tr>
<td>469</td>
<td>STOPPED</td>
</tr>
<tr>
<td>470</td>
<td>STARTED</td>
</tr>
<tr>
<td>471</td>
<td>XPLINK</td>
</tr>
<tr>
<td>472</td>
<td>SSL</td>
</tr>
<tr>
<td>473</td>
<td>CICSAPI</td>
</tr>
<tr>
<td>474</td>
<td>OPENAPI</td>
</tr>
<tr>
<td>475</td>
<td>EYUDA EXCI</td>
</tr>
</tbody>
</table>

---

**EYUDA general values in alphabetic order**

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

**EYUDA**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AASTERISK</td>
<td>165</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>ASSERTED</td>
<td>436</td>
</tr>
<tr>
<td>ASSOCIATIONS</td>
<td>293</td>
</tr>
<tr>
<td>AUTO</td>
<td>257</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

AUTOMATIC
   428
AUTOREGISTER
   427
AVG
   150
BACKOUTONLY
   179
BAPPL
   389
BASIC
   425
BATCHDI
   334
BCHLU
   335
BEFORE
   180
BELOW
   76
BLANK
   166
BLINK
   370
BLUE
   181
BOTH
   169
BRIDGE
   441
C
   182
CDSA
   79
CERTIFICATE
   426
CF
   385
CFTBL
   384
CHARSTR
   44
CHECK
   162
CICS
   87
CICSAPI
   473
CICSBTS
   442
Alphabetic EYUDA general values

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
COM7 401
COM8 402
CONACT 126
CONNDEF 275
CONNECTED 463
CONNECTING 380
CONSOLE 336
CONTINUE 270
CONTLU 337
CREATING 120
CTABL 97
Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<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>DISCON宁ING</td>
<td>383</td>
</tr>
<tr>
<td>DISCREETIONRY</td>
<td>462</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>DPL</td>
<td>438</td>
</tr>
<tr>
<td>DROP</td>
<td>110</td>
</tr>
<tr>
<td>DSA</td>
<td>78</td>
</tr>
<tr>
<td>DUPLICATE</td>
<td>143</td>
</tr>
<tr>
<td>DYNAM</td>
<td>258</td>
</tr>
<tr>
<td>ECDSA</td>
<td>83</td>
</tr>
<tr>
<td>ECI</td>
<td>451</td>
</tr>
<tr>
<td>EDSA</td>
<td>82</td>
</tr>
<tr>
<td>EJCODEF</td>
<td>423</td>
</tr>
<tr>
<td>EJDIJDEF</td>
<td>424</td>
</tr>
<tr>
<td>ELPA</td>
<td>86</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

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
EYUDA EXCI  475
FACILITY  431
FALSE  8
FENODDEF  415
FEPOODEF  413
FEPRODEF  416
FETRGDEF  414
FILE  193
FILEDEF  276
FIRSTREF  194
FORCE  107
## Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<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>HISTORY</td>
<td>459</td>
</tr>
<tr>
<td>HOTPOOL</td>
<td>452</td>
</tr>
<tr>
<td>HS</td>
<td>14</td>
</tr>
<tr>
<td>HTASK</td>
<td>454</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>IIOP</td>
<td>450</td>
</tr>
<tr>
<td>IIRQRECVR</td>
<td>448</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>Value</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>INVALIDFILE</td>
<td>456</td>
</tr>
<tr>
<td>IOERROR</td>
<td>455</td>
</tr>
<tr>
<td>JRNLDEF</td>
<td>277</td>
</tr>
<tr>
<td>JRNMDEF</td>
<td>278</td>
</tr>
<tr>
<td>JVM</td>
<td>453</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>LINK3270</td>
<td>430</td>
</tr>
<tr>
<td>LMS</td>
<td>205</td>
</tr>
<tr>
<td>LOCAL</td>
<td>111</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

LOGICAL
  206
LOGOFF
  207
LOGON
  60
LSTCMAS
  163
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
### Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXTASK</td>
<td>130</td>
</tr>
<tr>
<td>MCICS</td>
<td>64</td>
</tr>
<tr>
<td>MCONN</td>
<td>67</td>
</tr>
<tr>
<td>MDBX</td>
<td>66</td>
</tr>
<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>MIRROR</td>
<td>437</td>
</tr>
<tr>
<td>MIXIDPE</td>
<td>211</td>
</tr>
<tr>
<td>MJRNRL</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>
</tbody>
</table>
Alphabetic EYUDA general values

NEUTRAL 215
NEVER 265
NEW 216
NM 12
NO 2
NOCOPY 77
NOFORCE 217
NONE 108
NONVTAM 218
NOREENTPROT 435
NORMAL 22
NOSHUT 26
NOTAPPLIC 94
NOTCONNECTED 464
NOTOPEN 432
NOTWAIT 382
NULL 43
OFF 4
OLD 219
ON 3
ONCRPC 439
ONLY 220
OPEN 221
OPENAPI 474
OPID 222
OUTPUT 223
PARTDEF 281
Alphabetic EYUDA general values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<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>
<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>
</tbody>
</table>
Alphabetic EYUDA general values

PIPEDEF
467
PIPELINE
345
PLI
227
PMWINDOW
393
POOL
49
PRIMARY
141
PRINTER
228
PROCDEF
388
PROCESS
391
PROFDEF
283
PROGDEF
284
PROMPT
269
PRTNDEF
282
QUASIRENT
386
QUEUE
54
QUIESCE
140
QUIESCED
122
QUIESCING
48
RACFGID
41
RASINDSC
375
RBA
420
RDSA
156
READONLY
89
REALTIME
460
Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>REBUILD</td>
<td>378</td>
</tr>
<tr>
<td>RECONNECT</td>
<td>379</td>
</tr>
<tr>
<td>RECOVERY</td>
<td>230</td>
</tr>
<tr>
<td>RED</td>
<td>229</td>
</tr>
<tr>
<td>REENTPROT</td>
<td>434</td>
</tr>
<tr>
<td>REJECT</td>
<td>231</td>
</tr>
<tr>
<td>RELATED</td>
<td>263</td>
</tr>
<tr>
<td>RELEASE</td>
<td>304</td>
</tr>
<tr>
<td>RELEASESS</td>
<td>232</td>
</tr>
<tr>
<td>REMOTE</td>
<td>112</td>
</tr>
<tr>
<td>REMOVE</td>
<td>114</td>
</tr>
<tr>
<td>REMOVING</td>
<td>121</td>
</tr>
<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>RESUMING</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>RRS</td>
<td>447</td>
</tr>
<tr>
<td>RTADEF</td>
<td>273</td>
</tr>
<tr>
<td>RTRAN</td>
<td>51</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

RZSTTRPT 449
SAM 35
SCOPETYP 377
SCS 235
SCSPRINT 346
SDSA 154
SECONDARY 142
SECURITY 236
SEQTERM 410
SESSDEF 285
SHR 237
SHUT 25
SHUTDOWN 272
SIGNID 40
SIGNON 61
SKIP 238
SOCKET 446
SOSABOVE 159
SOSBELOW 158
SOSCDSA 133
SOSECDSA 135
SOSERDSA 136
SOSESDSA 138
SOSEUDSA 134
SOMVS 157
### Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOSRDSA</td>
<td>139</td>
</tr>
<tr>
<td>SOSSDSA</td>
<td>137</td>
</tr>
<tr>
<td>SOSUDSA</td>
<td>132</td>
</tr>
<tr>
<td>SPECIFIC</td>
<td>239</td>
</tr>
<tr>
<td>SSL</td>
<td>472</td>
</tr>
<tr>
<td>STALLED</td>
<td>131</td>
</tr>
<tr>
<td>STANDARD</td>
<td>332</td>
</tr>
<tr>
<td>START</td>
<td>116</td>
</tr>
<tr>
<td>STARTED</td>
<td>469</td>
</tr>
<tr>
<td>STARTIO</td>
<td>240</td>
</tr>
<tr>
<td>STARTUP</td>
<td>241</td>
</tr>
<tr>
<td>STAT</td>
<td>262</td>
</tr>
<tr>
<td>STATDEF</td>
<td>274</td>
</tr>
<tr>
<td>STOPPED</td>
<td>470</td>
</tr>
<tr>
<td>STRFIELD</td>
<td>242</td>
</tr>
<tr>
<td>SUM</td>
<td>147</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>30</td>
</tr>
<tr>
<td>SUSPENDED</td>
<td>457</td>
</tr>
<tr>
<td>SUSPENDING</td>
<td>458</td>
</tr>
<tr>
<td>SYSDEFAULT</td>
<td>243</td>
</tr>
<tr>
<td>SYSDUMP</td>
<td>128</td>
</tr>
<tr>
<td>SYSGROUP</td>
<td>104</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>28</td>
</tr>
</tbody>
</table>
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>TDQUEUE</td>
<td>443</td>
</tr>
<tr>
<td>TERMDEF</td>
<td>287</td>
</tr>
<tr>
<td>TERMINID</td>
<td>39</td>
</tr>
<tr>
<td>TERMINAL</td>
<td>245</td>
</tr>
<tr>
<td>TERMINATE</td>
<td>271</td>
</tr>
<tr>
<td>TERMSTART</td>
<td>444</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>
</tbody>
</table>
Alphabetic EYUDA general values

TURQUOISE 248
TWAIT 381
TWX 348
TYPTMDEF 291
U 249
UDSA 80
UNASSIGNED 109
UNCONDREL 250
UNDERLINE 372
UNKNOWN 433
UPDATEONLY 251
URIMPDEF 466
USER 88
USERID 42
USERPROG 349
UTABL 98
VALID 5
VALUE 33
VB 252
VELOCITY 461
VERIFY 253
VHS 15
VLS 9
VTAM 254
WAITING 47
WARMONLY 268
### Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEB</td>
<td>440</td>
</tr>
<tr>
<td>WEBSVDEF</td>
<td>468</td>
</tr>
<tr>
<td>XMRUN</td>
<td>445</td>
</tr>
<tr>
<td>XPLINK</td>
<td>471</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>
<tr>
<td>3270P</td>
<td>350</td>
</tr>
<tr>
<td>3270PRNT</td>
<td>406</td>
</tr>
<tr>
<td>3270TERM</td>
<td>405</td>
</tr>
<tr>
<td>3275</td>
<td>351</td>
</tr>
<tr>
<td>3277</td>
<td>352</td>
</tr>
<tr>
<td>3277CM</td>
<td>353</td>
</tr>
<tr>
<td>3284</td>
<td>354</td>
</tr>
<tr>
<td>3284CM</td>
<td>355</td>
</tr>
<tr>
<td>3286</td>
<td>356</td>
</tr>
<tr>
<td>3286CM</td>
<td>357</td>
</tr>
<tr>
<td>3600</td>
<td>358</td>
</tr>
<tr>
<td>3614</td>
<td>359</td>
</tr>
<tr>
<td>3650</td>
<td>360</td>
</tr>
<tr>
<td>3653</td>
<td>361</td>
</tr>
<tr>
<td>3767</td>
<td>362</td>
</tr>
<tr>
<td>3767C</td>
<td>363</td>
</tr>
<tr>
<td>3767I</td>
<td>364</td>
</tr>
<tr>
<td>3770</td>
<td>365</td>
</tr>
<tr>
<td>3770B</td>
<td>366</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA general values

<table>
<thead>
<tr>
<th>Value</th>
<th>EYUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>3770C</td>
<td>367</td>
</tr>
<tr>
<td>3770I</td>
<td>368</td>
</tr>
<tr>
<td>3790</td>
<td>369</td>
</tr>
</tbody>
</table>

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>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>VERSIONINVL</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 RESPONSE values in alphabetic order

This section lists the RESPONSE EYUDAs in alphabetic order.

<table>
<thead>
<tr>
<th>EYUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
</tr>
<tr>
<td>1032 BUSY</td>
</tr>
<tr>
<td>1042 DUPE</td>
</tr>
<tr>
<td>1030 ENVIRONERROR</td>
</tr>
<tr>
<td>1029 FAILED</td>
</tr>
<tr>
<td>1039 INCOMPATIBLE</td>
</tr>
<tr>
<td>1040 INUSE</td>
</tr>
<tr>
<td>1041 INVALIDATA</td>
</tr>
</tbody>
</table>
### Alphabetic EYUDA RESPONSE values

<table>
<thead>
<tr>
<th>Value</th>
<th>EYUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1036</td>
<td>INVALIDCMD</td>
</tr>
<tr>
<td>1028</td>
<td>INVALIDPARM</td>
</tr>
<tr>
<td>1027</td>
<td>NODATA</td>
</tr>
<tr>
<td>1034</td>
<td>NOTAVAILABLE</td>
</tr>
<tr>
<td>1026</td>
<td>NOTFOUND</td>
</tr>
<tr>
<td>1031</td>
<td>NOTPERMIT</td>
</tr>
<tr>
<td>1024</td>
<td>OK</td>
</tr>
<tr>
<td>1025</td>
<td>SCHEDULED</td>
</tr>
<tr>
<td>1033</td>
<td>SERVERGONE</td>
</tr>
<tr>
<td>1038</td>
<td>TABLEERROR</td>
</tr>
<tr>
<td>1035</td>
<td>VERSIONINVL</td>
</tr>
<tr>
<td>1037</td>
<td>WARNING</td>
</tr>
</tbody>
</table>

### EYUDA REASON values in numerical order

This section lists the REASON EYUDAs in numerical order.

<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>
</tbody>
</table>
Numerical EYUDA REASON values

1302  ECB
1303  SENTINEL
1304  FEEDBACK
1305  EVENT
1306  TOKEN
1307  MODIFY
1308  VIEW
1309  FIELDS
1310  ATTRIBUTE
1311  FROMCV
1312  TOCHAR
1313  FROMCHAR
1314  TOCV
1315  PARM
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
Numerical EYUDA REASON values

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
1359  EXCEPTION
1360  INVALIDEVT
1361  DATAERROR
1362  CMAS
1363  FIRST
1364  NEXT
1365  EXPIRED
1366  WORKLOAD
1367  ACTIONPARM
1368  CICSNAME
1369  MAXRECORDS
1370  QUERY

EYUDA REASON values in alphabetic order

This section lists the REASON EYUDAs in alphabetic order.

**EYUDA**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1331</td>
</tr>
<tr>
<td>1301</td>
</tr>
<tr>
<td>1367</td>
</tr>
<tr>
<td>1340</td>
</tr>
<tr>
<td>1338</td>
</tr>
<tr>
<td>1344</td>
</tr>
<tr>
<td>1329</td>
</tr>
</tbody>
</table>
Alphabetic EYUDA REASON values

1328   ARMNOTREG
1330   ARMPOLCHK
1310   ATTRIBUTE
1289   BACKWARD
1300   BY
1368   CICSNAME
1327   CICSRELBAD
1362   CMAS
1282   CONTEXT
1284   COUNT
1334   CPSMAPI
1339   CPSMSERVER
1332   CPSMSYSTEM
1333   CPSMVERSION
1299   CRITERIA
1361   DATAERROR
1319   DATALength
1291   DELAY
1302   ECB
1305   EVENT
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
Alphabetic EYUDA REASON values

<table>
<thead>
<tr>
<th>Code</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1285</td>
<td>LENGTH</td>
</tr>
<tr>
<td>1323</td>
<td>MAINTPOINT</td>
</tr>
<tr>
<td>1369</td>
<td>MAXRECORDS</td>
</tr>
<tr>
<td>1307</td>
<td>MODIFY</td>
</tr>
<tr>
<td>1364</td>
<td>NEXT</td>
</tr>
<tr>
<td>1358</td>
<td>NOSERVICE</td>
</tr>
<tr>
<td>1357</td>
<td>NOSTORAGE</td>
</tr>
<tr>
<td>1287</td>
<td>NOTFILTER</td>
</tr>
<tr>
<td>1292</td>
<td>NOTIFICATION</td>
</tr>
<tr>
<td>1335</td>
<td>NOTSUPPORTED</td>
</tr>
<tr>
<td>1336</td>
<td>NOTVSNCCONN</td>
</tr>
<tr>
<td>1281</td>
<td>OBJECT</td>
</tr>
<tr>
<td>1315</td>
<td>PARM</td>
</tr>
<tr>
<td>1316</td>
<td>PARMLEN</td>
</tr>
<tr>
<td>1341</td>
<td>PLEXMGR</td>
</tr>
<tr>
<td>1290</td>
<td>POSITION</td>
</tr>
<tr>
<td>1370</td>
<td>QUERY</td>
</tr>
<tr>
<td>1342</td>
<td>REQTIMEOUT</td>
</tr>
<tr>
<td>1295</td>
<td>RESOURCE</td>
</tr>
<tr>
<td>1283</td>
<td>RESULT</td>
</tr>
<tr>
<td>1294</td>
<td>SCOPE</td>
</tr>
<tr>
<td>1303</td>
<td>SENTINEL</td>
</tr>
<tr>
<td>1293</td>
<td>SIGNONPARM</td>
</tr>
<tr>
<td>1321</td>
<td>SOCRESOURCE</td>
</tr>
<tr>
<td>1322</td>
<td>SOERESOURCE</td>
</tr>
<tr>
<td>1320</td>
<td>SOLRESOURCE</td>
</tr>
<tr>
<td>1317</td>
<td>SUMOPT</td>
</tr>
<tr>
<td>1325</td>
<td>SYSLVLBAD</td>
</tr>
<tr>
<td>1324</td>
<td>SYSNOTACT</td>
</tr>
<tr>
<td>1326</td>
<td>SYSNOTLCL</td>
</tr>
<tr>
<td>1355</td>
<td>TASKDATAKEY</td>
</tr>
<tr>
<td>1280</td>
<td>THREAD</td>
</tr>
<tr>
<td>1297</td>
<td>TO</td>
</tr>
<tr>
<td>1312</td>
<td>TOCHAR</td>
</tr>
<tr>
<td>1314</td>
<td>TOCV</td>
</tr>
<tr>
<td>1306</td>
<td>TOKEN</td>
</tr>
<tr>
<td>1318</td>
<td>TYPE</td>
</tr>
</tbody>
</table>
## Alphabetic EYUDA REASON values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1345</td>
<td>USRID</td>
</tr>
<tr>
<td>1348</td>
<td>VERSION</td>
</tr>
<tr>
<td>1308</td>
<td>VIEW</td>
</tr>
<tr>
<td>1366</td>
<td>WORKLOAD</td>
</tr>
</tbody>
</table>
The CICS Transaction Server for z/OS library

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

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

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

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

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

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

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

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

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

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

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

CICS books for CICS Transaction Server for z/OS
General

- CICS Transaction Server for z/OS Program Directory, GI10-2586
- CICS Transaction Server for z/OS Migration from CICS TS Version 2.3, GC34-6425
Diagnosis

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

CICS family books

Communication

CICS Family: Interproduct Communication, SC34-6473
CICS Family: Communicating from CICS on System/390, SC34-6474

Licensed publications

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

CICS Diagnosis Reference, GC34-6899
CICS Data Areas, GC34-6902
CICS Supplementary Data Areas, GC34-6905
CICS Debugging Tools Interfaces Reference, GC34-6908

Other CICS books

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

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

Determining if a publication is current

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

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

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

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

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

You can perform most tasks required to set up, run, and maintain your CICSPlex SM system in one of these ways:
- using a 3270 emulator connected to CICSPlex SM
- using a 3270 emulator logged on to CICS
- using a 3270 emulator logged on to TSO
- using a 3270 emulator as an MVS system console
- using the CICSPlex SM web user interface.

IBM Personal Communications (Version 5.0.1 for Windows 95, Windows 98, Windows NT and Windows 2000; version 4.3 for OS/2) provides 3270 emulation with accessibility features for people with disabilities. You can use this product to provide the accessibility features you need in your CICSPlex SM system.
Index

A
ADDRESS command
description 12
summary of RESPONSE values 141
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 8
    with TPARSE 138
    with TRANSLATE 118

C
C programs
argument values for 3
CANCEL command
description 14
summary of RESPONSE values 141
CICS definitions
  creating 24
  removing 102
  updating 125
CICSPlex SM definitions
  creating 24
  removing 102
  updating 125
CICSPlex SM meta-data
  retrieving 52
CICSPlex SM notifications
  canceling 14
  requesting 62
COBOL programs
argument values for 2
command responses
summary 141
command-level interface
  specifying API commands 1
  specifying argument values 1
commands
ADDRESS 12
CANCEL 14
CONNECT 16
COPY 20
CREATE 24
DELETE 27
DISCARD 30
DISCONNECT 33
FEEDBACK 35

commands (continued)
  FETCH 39
  GET 46
  GETDEF 52
  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 136
  TERMINATE 117
  TPARSE 138
  TRANSLATE 118
  UNMARK 121
  UPDATE 125
commands, specifying
  using the command-level interface 1
  using the run-time interface 5
CONNECT command
description 16
summary of RESPONSE values 141
context
  changing default 88
  setting default 16
COPY command
description 20
summary of RESPONSE values 141
copying result set records 20
CREATE command
description 24
summary of RESPONSE values 141
CVDA values, translating 8

D
definitions, CICS
  creating 24
  removing 102
  updating 125
definitions, CICSPlex SM
  creating 24
  removing 102
  updating 125
DELETE command
description 27
summary of RESPONSE values 142
deleting result set records 27
DISCARD command
  description 30
  summary of RESPONSE values 142
DISCONNECT command
  description 33
  summary of RESPONSE values 142

E
ECB field
  requesting 12
  event control block (ECB)
    requesting 12
EYU9XESV security routine
  options on CONNECT 17
EYUAPI function
  description 130
EYUDA values
  summary of 149
  translating 8
EYUNINIT function
  description 131
EYUREAS function
  description 132
EYURESAP function
  description 133
EYUTERM function
  description 134
EYUVALUE function
  description 8

F
FEEDBACK command
  description 35
  summary of RESPONSE values 142
FETCH command
  description 39
  summary of RESPONSE values 142
  filter
    defining 111
    discarding 30
  filter expression
    specifying
      on GET 47
      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 46
  summary of RESPONSE values 143
GETDEF command
  description 52
  summary of RESPONSE values 143
GROUP command
  description 58
  (continued)
  summary of RESPONSE values 143

L
language considerations
  general 8
  length options, specifying 9
LISTEN command
  description 62
  summary of RESPONSE values 143
LOCATE command
  description 65
  summary of RESPONSE values 143

M
MARK command
  description 70
  summary of RESPONSE values 144
  marking result set records 70
  meta-data, CICSPlex SM
    retrieving 52
  modification expression
    specifying
      on SET 106
      on UPDATE 125
  MVS restrictions 8

N
notifications, CICSPlex SM
  canceling 14
  requesting 62

O
ORDER command
  description 74
  summary of RESPONSE values 144
  order expression
    specifying
      on ORDER 74
      on SPECIFY VIEW 114

P
parameter expression
  specifying
    on CREATE 24
    on GET 47
    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 144
PERFORM SET command  
description 83  
summary of RESPONSE values 144  
PL/I programs  
argument values for 3

Q  
QUALIFY command  
description 88  
summary of RESPONSE values 145

QUERY command  
description 91  
summary of RESPONSE values 145

R  
REASON option  
description 9  
summary of values 141  
RECEIVE command  
description 94  
summary of RESPONSE values 145  
REFRESH command  
description 97  
summary of RESPONSE values 145  
refreshing result set records 97  
REMOVE command  
description 102  
summary of RESPONSE values 145  
resource table  
translating attributes  
with EYUVALUE 8  
with TPARSE 138  
with TRANSLATE 118  
RESPONSE option  
description 9  
summary of values 141  
responses, command  
summary 141  
result set commands  
COPY 20  
DELETE 27  
DISCARD 30  
FETCH 39  
GET 46  
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 30  
querying 91  
result set (continued)  
records  
copying 20  
deleting 27  
marking 70  
refreshing 97  
retrieving meta-data 52  
REXX commands  
TBUILD 136  
TPARSE 138  
REXX functions  
EYUAPI 130  
EYUNIT 131  
EYUREAS 132  
EYURESP 133  
EYUTERM 134  
REXX run-time interface  
commands 135  
specifying API commands 5  
specifying argument values 5

S  
scope  
changing default 88  
setting default 16  
security  
options on CONNECT 17  
sentinel field  
requesting 12  
SET command  
description 105  
summary of RESPONSE values 146  
SPECIFY FILTER command  
description 111  
summary of RESPONSE values 146  
SPECIFY VIEW command  
description 114  
summary of RESPONSE values 146  
summarized result set  
creating 58  
summary expression  
specifying 59  
summary options  
specifying 59  
syntax diagrams, reading 7

T  
TBUILD command  
description 136  
TERMINATE command  
description 117  
summary of RESPONSE values 146  
TPARSE command  
description 138  
TRANSLATE command  
description 118  
summary of RESPONSE values 146
translating
  resource table attributes
    with EYUVALUE  8
    with TPARSE  138
    with TRANSLATE  118

U
UNMARK command
  description  121
  summary of RESPONSE values  147
UPDATE command
  description  125
  summary of RESPONSE values  147

V
view
  building  114
  discarding  30
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.
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

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

Adobe and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Other product and service names might be trademarks of IBM or other companies.
Sending your comments to IBM

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

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

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

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

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

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

• By mail, to this address:

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

• By fax:
  – From outside the U.K., after your international access code use 44–1962–816151
  – From within the U.K., use 01962–816151

• Electronically, use the appropriate network ID:
  – IBMLink: HURSLEY(IDRCF)
  – Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:
• The publication title and order number
• The topic to which your comment applies
• Your name and address/telephone number/fax number/network ID.