

CICS® Transaction Server for OS/390®



CICSplex® SM Application Programming Guide

Release 3

CICS® Transaction Server for OS/390®



CICSplex® SM Application Programming Guide

Release 3

Note!

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

First Edition, March 1999

This edition applies to Release 3 of CICS Transaction Server for OS/390, program number 5655-147, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Information in this edition was previously contained in SC33-1430-02, which is now obsolete. Make sure you are using the correct edition for the level of the product.

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

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

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

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

© **Copyright International Business Machines Corporation 1995, 1999. All rights reserved.**

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Sample programs	viii
Programming interface information	viii
Trademarks	viii
Preface	ix
Who this book is for	ix
What you need to know	ix
How to use this book	ix
Notes on terminology	ix
CICS System Connectivity	x
Bibliography	xiii
CICS Transaction Server for OS/390	xiii
CICS books for CICS Transaction Server for OS/390	xiii
CICSplex SM books for CICS Transaction Server for OS/390	xiv
Other CICS books	xiv
Books from related libraries	xiv
Summary of Changes	xvii
Major changes to this book include the following:	xvii
Chapter 1. An overview of the CICSplex SM API	1
Supported environments and languages	1
Available interfaces	1
Connecting to CICSplex SM	2
The connection process	4
Security considerations	5
Compatibility between environments	6
Compatibility between releases of CICSplex SM	7
Migrating applications to a new release	8
Accessing resource tables from a new release	8
Accessing resource tables from a previous release	9
Sample programs	11
Where to find more information	11
Chapter 2. Using the CICSplex SM API	13
CICSplex SM managed objects	13
Types of managed objects	13
CICSplex SM resource tables	15
Building a customized resource table record	16
How to create copybooks for customized resource table records	17
Selecting managed objects	19
Setting the context and scope	19
Using filter expressions	20
Working with result sets	23
An overview of result set commands	23
Retrieving records from a result set	26
Positioning the record pointer in a result set	28
Processing selected records in a result set	29

Summarizing the records in a result set	32
Sorting the records in a result set	34
Modifying managed resources	35
Modifying resource attributes	36
Performing an action against a resource	37
Working with CICSplex SM and CICS definitions	37
Asynchronous processing	39
Using the LISTEN command	41
Using the NOWAIT option	41
Using tokens to identify a request	42
Using the ADDRESS command	42
Using the RECEIVE command	43
Using CICSplex SM tokens	44
Using meta-data resource tables	44
ATTR	45
ATTRAVA	49
METADESC	49
OBJECT	50
OBJECT	51
Querying the CICSplex SM API exit	53
Chapter 3. Writing an EXEC CPSM program	55
Using the resource table copy books	55
How to access the copy books	55
Copybook names and aliases	55
Copybook format	56
Copybook data characteristics	56
Supplied copy books	57
Language and environment considerations	68
Assembler H considerations	68
PL/I considerations	68
NetView considerations	69
User-replaceable programs	69
CICS Global User exit programs	69
Status programs	69
Translating your program	70
Specifying the CPSM translator option	70
Compiling your program	71
Assembler H considerations	71
PL/I considerations	72
COBOL considerations	72
C/370 considerations	72
Link editing your program	73
Assembler H considerations	73
PL/I, VS COBOL II, and C/370 considerations	74
Run-time considerations	74
Chapter 4. Dealing with exception conditions	77
Default CICSplex SM exception handling	77
Using the RESPONSE and REASON options	77
Types of responses	77
Testing for RESPONSE and REASON	80
Retrieving FEEDBACK records	81
Using the FEEDBACK command	81

Evaluating a FEEDBACK record	82
Availability of FEEDBACK records	84
An example of FEEDBACK for a result set	85
Additional processing for BAS	86
Evaluating error result set records	86
Evaluating BINSTERR resource table records	86
Evaluating BINCONRS resource table records	87
Evaluating BINCONSC resource table records	88
An example of a BAS error result set	89
Chapter 5. Writing a REXX program	91
Accessing the API environment	91
Specifying an API command	92
Accessing resource table data	93
Translating attribute values	94
Processing CHANGETIME and CREATETIME attributes	95
Chapter 6. REXX error handling	97
Translation errors	97
Run-time errors	98
TPARSE and TBUILD errors	98
Messages	98
EYU_TRACE data	98
Appendix A. BINCONRS, BINCONSC, and BINSTERR error codes	101
BINCONRS	101
BINCONSC	101
BINSTERR	102
Appendix B. Sample program listings	103
Sample program EYU#API1	104
Sample program EYUCAPI2	108
Sample program EYUAAPI3	115
Sample program EYULAPI4	134
Glossary	147
Index	159
Sending your comments to IBM	161

contents

Notices

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

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

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

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

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Sample programs

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

Programming interface information

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

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

Trademarks

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

CICS	MVS/ESA
CICS OS/2	NetView
CICS/ESA	OS/2
CICS/MVS	OS/390
CICS/VSE	RACF
CICSplex	S/370
C/370	S/390
DB2	SP
IBM	System/390
IMS	VSE/ESA
IMS/ESA	VTAM

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

Preface

This book provides programming information for the IBM® CICSplex® System Manager (CICSplex SM) element of CICS® Transaction Server for OS/390® Release 3. It describes how to use the application programming interface (API) to access CICSplex SM data and services.

Who this book is for

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

What you need to know

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

For guidance information on how to use the CICSplex SM API see the *CICSplex SM Application Programming Reference*.

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

CICSplex SM Managing Business Applications

For information about Business Application Services definitions.

CICSplex SM Managing Resource Usage

For information about real-time analysis and Monitoring definitions.

CICSplex SM Managing Workloads

For information about Workload Manager definitions.

How to use this book

This book provides guidance information for the CICSplex SM API.

It introduces the API, describes the various environments it supports, and provides examples of its use. If this is your first experience with the API, it will probably help to read through the guide more or less from start to finish.

Notes on terminology

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

Other terms used in this book are:

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

CICS System Connectivity

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

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

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

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

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

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

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

Table 1 on page xi shows which CICS systems may be directly connected to which releases of CICSplex SM.

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

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

Bibliography

CICS Transaction Server for OS/390

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

CICS books for CICS Transaction Server for OS/390

General

<i>CICS Master Index</i>	SC33-1704
<i>CICS User's Handbook</i>	SX33-6104
<i>CICS Glossary</i> (softcopy only)	GC33-1705

Administration

<i>CICS System Definition Guide</i>	SC33-1682
<i>CICS Customization Guide</i>	SC33-1683
<i>CICS Resource Definition Guide</i>	SC33-1684
<i>CICS Operations and Utilities Guide</i>	SC33-1685
<i>CICS Supplied Transactions</i>	SC33-1686

Programming

<i>CICS Application Programming Guide</i>	SC33-1687
<i>CICS Application Programming Reference</i>	SC33-1688
<i>CICS System Programming Reference</i>	SC33-1689
<i>CICS Front End Programming Interface User's Guide</i>	SC33-1692
<i>CICS C++ OO Class Libraries</i>	SC34-5455
<i>CICS Distributed Transaction Programming Guide</i>	SC33-1691
<i>CICS Business Transaction Services</i>	SC34-5268

Diagnosis

<i>CICS Problem Determination Guide</i>	GC33-1693
<i>CICS Messages and Codes</i>	GC33-1694
<i>CICS Diagnosis Reference</i>	LY33-6088
<i>CICS Data Areas</i>	LY33-6089
<i>CICS Trace Entries</i>	SC34-5446
<i>CICS Supplementary Data Areas</i>	LY33-6090

Communication

<i>CICS Intercommunication Guide</i>	SC33-1695
<i>CICS Family: Interproduct Communication</i>	SC33-0824
<i>CICS Family: Communicating from CICS on System/390</i>	SC33-1697
<i>CICS External Interfaces Guide</i>	SC33-1944
<i>CICS Internet Guide</i>	SC34-5445

Special topics

<i>CICS Recovery and Restart Guide</i>	SC33-1698
<i>CICS Performance Guide</i>	SC33-1699
<i>CICS IMS Database Control Guide</i>	SC33-1700
<i>CICS RACF Security Guide</i>	SC33-1701
<i>CICS Shared Data Tables Guide</i>	SC33-1702
<i>CICS Transaction Affinities Utility Guide</i>	SC33-1777
<i>CICS DB2 Guide</i>	SC33-1939

CICSplex SM books for CICS Transaction Server for OS/390

General

<i>CICSplex SM Master Index</i>	SC33-1812
<i>CICSplex SM Concepts and Planning</i>	GC33-0786
<i>CICSplex SM User Interface Guide</i>	SC33-0788
<i>CICSplex SM View Commands Reference Summary</i>	SX33-6099

Administration and Management

<i>CICSplex SM Administration</i>	SC34-5401
<i>CICSplex SM Operations Views Reference</i>	SC33-0789
<i>CICSplex SM Monitor Views Reference</i>	SC34-5402
<i>CICSplex SM Managing Workloads</i>	SC33-1807
<i>CICSplex SM Managing Resource Usage</i>	SC33-1808
<i>CICSplex SM Managing Business Applications</i>	SC33-1809

Programming

<i>CICSplex SM Application Programming Guide</i>	SC34-5457
<i>CICSplex SM Application Programming Reference</i>	SC34-5458

Diagnosis

<i>CICSplex SM Resource Tables Reference</i>	SC33-1220
<i>CICSplex SM Messages and Codes</i>	GC33-0790
<i>CICSplex SM Problem Determination</i>	GC33-0791

Other CICS books

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

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

Books from related libraries

Books in related libraries include:

IBM CICS for MVS/ESA Version 4.1

<i>Application Programming Guide</i> , SC33-1169
<i>Application Programming Reference</i> , SC33-1170
<i>System Programming Reference</i> , SC33-1171

IBM CICS TS for OS/390 Release 1

<i>CICS Application Programming Guide</i> , SC33-1687
<i>CICS Application Programming Reference</i> , SC33-1688
<i>CICS System Programming Reference</i> , SC33-1689
<i>CICS Distributed Transaction Programming Guide</i> , SC33-1691
<i>CICS Front End Programming Interface User's Guide</i> , SC33-1692

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

TSO/E Version 2

Programming Guide, SC28-1874
Programming Services, SC28-1875
REXX/MVS User's Guide, SC28-1882
REXX/MVS Reference, SC28-1883

NetView Version 2.4

Application Programming Guide, SC31-7081
RODM Programming Guide, SC31-7095
Customization: Using Assembler, SC31-7094
Customization: Using PL/I and C, SC31-7093

Assembler H Version 2

Application Programming Guide, SC26-4036
Application Programming Language Reference, GC26-4037

VS COBOL II Version 1.3.2

Application Programming Guide for MVS and CMS, SC26-4045
Application Programming Guide for VSE, SC26-4697
Application Programming Language Reference, GC26-4047
Application Programming Debugging, SC26-4049

PL/I Version 2.3

Programming Language Reference, SC26-4308
Optimizing Compiler Programmer's Guide, SC33-0006

IBM C/370 Version 2.1

Programming Guide, SC09-1384
Reference Summary, SC09-1211

Summary of Changes

This book is based on information from the *CICSplex System Manager Application Programming Interface Release 3*, SC33-1430-02. The information in this book has been updated to incorporate changes made for CICSplex SM for CICS Transaction Server for OS/390 Release 3. Changes made since the last edition are indicated by vertical bars to the left of the change.

Major changes to this book include the following:

The following additions and changes have been made to the functions of CICSplex SM for CICS Transaction Server for OS/390 Release 3.

- The information in this book derives from the old *CICSplex System Manager Application Programming Interface* which has been split into two books the other being the *CICSplex SM Application Programming Reference*.
- The Resource Table Summary has been moved to the *CICSplex SM Resource Tables Reference*.
- Chapter 4, "Dealing with exception conditions" on page 77 has been reorganized and expanded and now includes new data in Table 7 on page 83 and a new section "An example of a BAS error result set" on page 89.
- There is a new Appendix A, "BINCONRS, BINCONSC, and BINSTERR error codes" on page 101.

Chapter 1. An overview of the CICSplex SM API

The CICSplex SM application programming interface (API) provides you with access to CICS system management information and allows you to invoke CICSplex SM services from an external program. The API can provide a single interface for programs designed to monitor and control the CICS systems in your enterprise. In addition, the API provides an interface to CICSplex SM itself. So you can also write programs to access the administrative functions that control the way CICSplex SM operates.

Some typical uses of the API include:

- Monitoring key resources in your CICS environment.
- Changing the status of CICS resources relative to other conditions in your enterprise.
- Controlling the flow of change to your CICS environment.
- Passing the information provided by CICSplex SM to an automation product.
- Developing alternative display and report formats for CICS and CICSplex SM data.
- Processing CICSplex SM notifications about events such as:
 - Real-time analysis thresholds being reached
 - Operational state changes in resources being reported to the NetView® Resource Object Data Manager (RODM) facility
- Creating and maintaining CICSplex SM definitions for Business Application Services, for workload management, real-time analysis, and resource monitoring.
- Creating and maintaining CICS resource definitions in the CICSplex SM data repository.

Supported environments and languages

The API can be called from programs running in a variety of environments:

- MVS/ESA batch
- MVS/ESA TSO
- MVS/ESA NetView
- CICS/ESA®.

Note: The CICSplex SM API cannot be called from within a NetView RODM method. For details on the restrictions that apply to RODM method services, see the *NetView RODM Programming Guide*.

Available interfaces

CICSplex SM provides two interfaces for API users:

Command-level interface

This interface uses the CICS translator to accept EXEC CPSM statements and translate them into the appropriate sequence of instructions in the source

connecting to CICSPlex SM

language. These instructions are then linked to an interface stub routine that is supplied by CICSPlex SM.

The command-level interface is available for programs written in the following languages:

- Assembler H Version 2 and later
- OS PL/I Optimizing Compiler Version 2.3 and later
- VS COBOL II Compiler Version 1.3.2 and later
- C/370™ Version 2.1. and later

Table 2 shows which languages are supported by the command-level interface in each environment.

Table 2. Programming languages supported by the command-level interface

Environment	Assembler	COBOL	PL/I	C
CICS/ESA	√	√	√	√
MVS/ESA Batch	√	√	√	√
MVS/ESA TSO	√	√	√	√
MVS/ESA NetView	√		√	√

Run-time interface

The run-time interface supports programs written as REXX EXECs in the following MVS/ESA environments:

- Batch
- TSO
- NetView.

This interface consists of a REXX function package that is supplied by CICSPlex SM. The function package accepts commands in the form of text strings and generates the appropriate API calls.

Connecting to CICSPlex SM

You can think of a CICSPlex SM API program as existing in or having access to three environments:

User environment

The program itself and the environment in which it runs, such as MVS/ESA or CICS.

CICSPlex SM environment

The data that CICSPlex SM maintains and the services it provides to the program.

Managed resource environment

The resources that CICSPlex SM manages and which the program can access.

Before your program can access the CICSPlex SM environment and the resources it manages, you must establish a connection to CICSPlex SM. This connection is called an API processing thread and serves two basic purposes:

- When a thread is created, the user is identified so that security validation and auditing of the program's operations can take place transparently.

- There are implicit relationships between some API functions, and those relationships are maintained at the thread level. Each thread is considered a unique API user and no resources can cross the boundary of a thread.

Once a thread is created, your program can issue commands within a context that includes:

- The local CMAS in the MVS/ESA image where the program runs.
- Any CICSplex that is managed by the local CMAS.
- Any CMAS with an active communication link to the local CMAS.

A simple API program would establish only a single thread. You could establish the thread, perform the desired operations, and then terminate the thread. A more complex program might maintain several concurrent threads to perform parallel operations that would be prohibited on a single thread or to simplify the correlation of commands and results.

You can use the following commands to manage an API thread:

- CONNECT** Establishes a connection to CICSplex SM, defines an API processing thread, and provides default settings for the thread. The thread is maintained by the CMAS that is supporting your API session.
- DISCONNECT** Disconnects an API processing thread from CICSplex SM and releases any resources associated with the thread.
- QUALIFY** Defines the CICSplex SM context and scope for subsequent commands issued by the thread.
- TERMINATE** Terminates all API processing on all the threads created by the CICS or MVS/ESA task that issues the command.

These commands manage the connection between the user environment (your program) and CICSplex SM; they do not affect the managed resources. Figure 1 on page 4 illustrates the impact these commands have on the API environment.

connecting to CICSplex SM

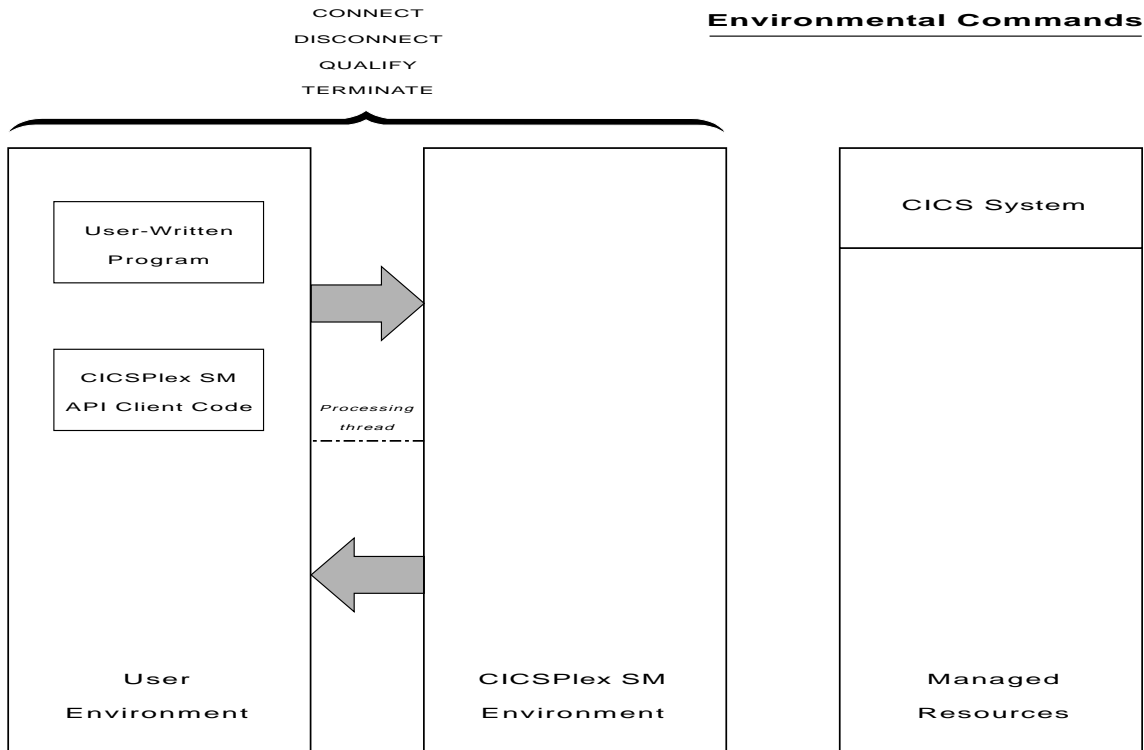


Figure 1. API commands involved in managing a thread

For complete descriptions of these commands, see *CICSplex SM Application Programming Reference*.

The connection process

The process of connecting to CICSplex SM varies according to what type of program you write and where it runs. For programs written using the command-level interface, keep in mind the following requirements:

CICS

A program written to run as a CICS application must be linked with the proper stub routine and must run in a CICS system that is being actively managed by CICSplex SM as a local MAS.

A connection is established first to the MAS agent code that resides in the CICS system and then to the CMAS that controls that MAS. On the CONNECT command, you can specify a CONTEXT of:

- The CMAS that is being connected to
- Any CICSplex that is managed by that CMAS
- Any CMAS with an active communication link to that CMAS

Batch, NetView, or TSO

A program written to run as a batch job or under NetView or TSO must be linked with the proper stub routine and must run in the same MVS image as the CMAS to which you want to connect.

In these environments, if there is more than one CMAS in the MVS/ESA image, the API selects a suitable CMAS and establishes a connection. The following rules apply to the selection of a CMAS:

- The CMAS must be running the same version of CICSplex SM as the run-time module (EYU9AB00).
- If the context specified on the CONNECT command is a CMAS, CICSplex SM connects to that CMAS. If that CMAS is either not active or not running the appropriate version of CICSplex SM, the CONNECT command fails.
- If the context specified on CONNECT is a CICSplex, CICSplex SM selects a CMAS running the appropriate version that participates in the management of the CICSplex.
- If no context is specified on CONNECT, CICSplex SM connects to the CMAS that was most recently started, provided it is running the appropriate version of CICSplex SM.

The CICSplex SM API also supports another type of batch environment. A program can issue API commands from an address space that is running a CICS system without itself being a CICS transaction. In other words, the program can run as a separate MVS/ESA task in the same address space as the CICS system. This type of program must be linked with the batch environment stub routine and the connection process is the same as for other batch programs.

Note: A program that is a CICS transaction must be run in a CICS system that is a CICSplex SM MAS.

For details on the stub routines that are required for each of these environments, see “Link editing your program” on page 73.

Note: For programs written in REXX, the connection process is the same as for a command-level program that runs in the same environment (batch, TSO, or NetView). No stub routine is required, but the REXX function package that is supplied by CICSplex SM must have been properly installed.

Security considerations

When an API program requests a connection to CICSplex SM, the CMAS being connected to attempts to extract user authorization data from the environment. How the connection is established depends upon whether such authorization data exists and whether security is active in the CMAS:

If security exists

Regardless of whether CMAS security is active, if a security environment exists where the API program is running:

- The API security routine, EYU9XESV, is not called.
- The USER and SIGNONPARM options on the CONNECT command are ignored.
- The API program is connected with the user ID of the invoking user, as obtained from the accessor environment element (ACEE).

Note: If CMAS security is not active, the ACEE user ID is not validated by CICSplex SM.

This type of security environment may exist when a program runs under TSO, batch, NetView, or a local MAS where CICS security is active. Security checking is performed by the environment where the API program is running.

If security does not exist and CMAS security is not active

- The API security routine, EYU9XESV, is not called.
- The USER and SIGNONPARM options on the CONNECT command are ignored.
- No signon is performed. However, the user ID specified in the XESV_CONN_USERID field of the security routine parameter block, EYUBXESV, is associated with the connection.

This type of security environment may exist when a program runs under a local MAS where CICS security is not active. Since CMAS security is not active, no security checking is performed.

If security does not exist and CMAS security is active

- The API security routine, EYU9XESV, is called.
- The USER and SIGNONPARM values from the CONNECT command are passed to EYU9XESV.
- A signon is performed using the user ID returned by EYU9XESV, but no password checking is performed. By default, EYU9XESV returns the default CICS user ID for the CMAS (the DFLT_UID value).

This type of security environment may exist when a program runs under a local MAS where CICS security is not active. Since CMAS security is active, security checking is performed by EYU9XESV.

Table 3 summarizes the levels of API security and the conditions under which they are implemented.

Table 3. Possible API security environments

Environment Security	CMAS Security	No CMAS Security
YES	EYU9XESV not called. CONNECT options ignored. User ID=ACEE.	EYU9XESV not called. CONNECT options ignored. User ID=ACEE (not checked).
NO	EYU9XESV called. CONNECT options passed. User ID=As returned by EYU9XESV (signon with no password checking).	EYU9XESV not called. CONNECT options ignored. User ID= XESV_CONN_USERID (no signon).

For a description of the USER and SIGNONPARM options, see the API CONNECT Command in the *CICSplex SM Application Programming Reference*. For a description of EYU9XESV and information on customizing this security routine, see *CICS RACF Security Guide*.

Compatibility between environments

Once you have written a CICSplex SM API program to run in one environment, you can take that program and run it in another environment with only minor modifications.

For example, if you want to take a CICS application written with EXEC CPSM commands and convert it to an MVS/ESA batch program, you should:

- Make the appropriate code changes, such as:
 - Remove any EXEC CICS commands that may be included
 - Add the necessary MVS calls
- Relink-edit the program with the batch environment stub routine.

Note: A REXX program can be moved from one MVS/ESA environment (batch, TSO, or NetView) to another without modification, provided you have not used any environment-specific functions.

Before you try to move an EXEC CPSM program to an environment other than the one for which it was written, you should review the following sections:

- “Language and environment considerations” on page 68
- “Translating your program” on page 70
- “Link editing your program” on page 73.

Compatibility between releases of CICSplex SM

Once you have written an API program to run under one release of CICSplex SM, you can continue to access the data provided by that release, or you can access the data available from a later release of the product. In general, if you plan to access more than one release of the CICSplex SM API, keep the following in mind:

Run-time environment

The run-time version of a CICSplex SM API program is equal to the level of the CMAS to which it connects:

- For a program written to run as a CICS application, the run-time version is that of the CMAS to which the MAS is connected.
- For a program written to run as a batch job or under NetView or TSO, the version is determined by the version of the CICSplex SM run-time module (EYU9AB00), which is distributed in the version's SEYUAUTH library.

The run-time version of a program must be greater than or equal to:

- The version of the stub routine module (EYU9AxSI) with which the program was link edited.
 - For CICS programs, the stub module is called EYU9AMSI and is distributed in the version's SEYULOAD library.
 - For batch, TSO, or NetView programs, the module is called EYU9ABSI and is distributed in the version's SEYUAUTH library.

In addition, the version of the stub module for any separately link edited and called programs must be the same as the version used to link edit the program that issued the CONNECT command.

- The value specified on the VERSION option of the CONNECT command.

Note: For programs written in REXX, the run-time version must be greater than or equal to the version of the function package (EYU9AR00), which is distributed in the version's SEYUAUTH library.

VERSION option

The VERSION option on the CONNECT command controls which release of CICSplex SM resource tables are available to your program (resource tables are the external representation of CICSplex SM data).

compatibility between releases

- An API program cannot access data from a release of CICSplex SM earlier than Release 2 (the release in which the API was introduced). The VERSION value must be set to 0120 or greater.
- An API program cannot access data from a release of CICSplex SM later than the run-time module that you specify. The VERSION value must be less than or equal to the release of the run-time module.
- An API program can access data from a later release of CICSplex SM than that which the program was originally written for, provided:
 - You compile your program using the appropriate copy books for the version specified.
 - Your program is compatible with the copy books for the version specified.

CONTEXT option

The CONTEXT option that is supported by various API commands determines which CICS systems your program receives data from. The CONTEXT value can be set to any CMAS or CICSplex running any currently supported release of CICSplex SM. Note, however, that the release level of the CMAS or CICSplex must be the same as the release of the run-time module.

CURRENT option

When specifying the CURRENT option, the record pointer does not move (that is, a subsequent FETCH retrieves the same record). Previously, the record pointer moved to the next record. For further information, see “Positioning the record pointer in a result set” on page 28.

Migrating applications to a new release

In order to migrate your application programs to the new release so that they can benefit from the full function available there, see “Accessing resource tables from a new release.” If you need to continue to run application programs in an earlier release CICSplex SM environment whilst being able to manage this environment using the services provided in the new release, you must first read the sections on CICSplex SM migration in the *CICS Transaction Server for OS/390: Migration Guide*.

Accessing resource tables from a new release

You can access the most up-to-date CICSplex SM resource tables by running an existing program under a new release of the API.

Note: To take full advantage of a new CICSplex SM function (such as Business Application Services), however, you would have to modify an existing program or create a new one.

To run an existing API program under a new release of CICSplex SM:

- Make sure the following are available to your program:
 - The run-time module for the new release (EYU9AB00 from the new release's SEYUAUTH library)
 - A CMAS that is running the new release
- Change the VERSION value on the CONNECT command to reflect the new release of CICSplex SM (for example, 0140 for CICS Transaction Server for

OS/390 Release 3) and relink-edit the program using the stub module supplied in the new release.

- Review the possible effects of any changes to the CICSplex SM resource tables.

Attributes may be added to a resource table in a new release, which could affect your program's references to that table. And with the addition or modification of attributes, the length of a given resource table may change from one release to another. The resource table copy books that are distributed with the new release are a good source of information about such changes.

If your program receives RESPONSE and REASON values of INVALIDPARM LENGTH when you run it under a new release of CICSplex SM, the table length may have increased and your data buffer may not be long enough to accommodate the new resource table records.

- If you are using customized views of resource tables, you are advised to check that the names of any new resource tables do not duplicate the names of your customized views, as this could affect your processing. For further details, see "Building a customized resource table record" on page 16 .

For a complete list of new and changed resource tables in a given release, refer to *CICSplex SM Resource Tables Reference*.

Accessing resource tables from a previous release

To continue accessing the resource tables supplied with a previous release of CICSplex SM:

- Specify the release of CICSplex SM data that you want to access on the VERSION option of the CONNECT command.
- Use the run-time module (EYU9AB00) supplied with the release you want to access or a subsequent release that supports it.
- Use a version of the stub module (EYU9AxSI) that is less than or equal to the run-time module.

Table 4 on page 10 illustrates valid combinations of the VERSION option, stub module and run-time module for accessing data from different releases of CICSplex SM.

compatibility between releases

Table 4. Valid ways to access data from different releases

VERSION value	Stub module (EYU9AxSI)	Run-time module (EYU9AB00)	CMASs available	CMAS used	Data available
0120	V1R2	V1R2	V1R2	V1R2	V1R2
0120	V1R2	V1R3	V1R3	V1R3	V1R2
0120	V1R3	V1R3	V1R2 V1R3	V1R3	V1R2
0120	V1R4	V1R4	V1R2 V1R3 V1R4	V1R2	V1R2
0130	V1R3	V1R3	V1R2 V1R3	V1R3	V1R3
0130	V1R3	V1R4	V1R2 V1R3 V1R4	V1R4	V1R3
0130	V1R4	V1R4	V1R2 V1R3 V1R4	V1R4	V1R3
0140	V1R4	V1R4	V1R2 V1R3 V1R4	V1R4	V1R4

Table 5 shows some invalid combinations of the VERSION option, run-time module, and stub module and describes why they produce an error.

Table 5. Common errors in accessing different releases

VERSION value	Stub module (EYU9AxSI)	Run-time module (EYU9AB00)	CMASs available	Error description
0120	V1R3	V1R2	V1R2 V1R3	Stub module release level is greater than run-time module.
0130	V1R3	V1R2	V1R2 V1R3	Stub module release level is greater than run-time module.
0130	V1R2	V1R2	V1R2	VERSION value is greater than run-time module.
0130	V1R3	V1R3	V1R2	No CMAS available at the required run-time level.

Note: For programs written in REXX, the compatibility issues are similar. The Release 2 function package (which contains the necessary stub module) can run successfully with either the Release 2 or Release 3 run-time module. The Release 3 function package, however, cannot run with the Release 2 run-time module; the Release 3 module is required.

Sample programs

Sample programs for each supported language are distributed with CICSplex SM in source form. These samples are provided to illustrate the types of programs you can write and the commands you need to use in those programs.

The sample programs are distributed in members called EYUxAPI n , where x is a 1-character language identifier and n is a sequential program identifier. For example, EYUCAPI1 is sample program number 1 coded in C.

The names of the sample programs and the libraries where they can be found are shown in the following table.

Table 6. Sample programs provided with CICSplex SM

Language	Programs	Library
Assembler	EYUAAPI1 EYUAAPI2 EYUAAPI3	SEYUSAMP
COBOL	EYULAPI1 EYULAPI2 EYULAPI4	SEYUCOB
PL/I	EYUPAPI1 EYUPAPI2	SEYUPL1
C	EYUCAPI1 EYUCAPI2	SEYUC370
REXX	EYU#API1 EYU#API2 EYU#API3	SEYUCLIB

A listing is provided for each sample program (in one of its supported languages) in Appendix B, "Sample program listings" on page 103.

Note: Additional sample CICSplex SM API programs are available on the World Wide Web at:

<http://www.software.ibm.com/ts/cicsplex/>

Where to find more information

This book is divided into three parts. Background information on the CICSplex SM API and guidance for writing an API program are contained in the following chapter in part one:

- Chapter 2, "Using the CICSplex SM API" on page 13 introduces a variety of concepts that you need to understand before you use the API.

Part two deals with EXEC CPSM programming considerations:

- Chapter 3, "Writing an EXEC CPSM program" on page 55 describes how to write an API program using the command-level interface.
- Chapter 4, "Dealing with exception conditions" on page 77 describes the tools and techniques you can use to handle errors in a CICSplex SM API program.

Part three deals with REXX programming considerations:

where to find more information

- Chapter 5, “Writing a REXX program” on page 91 describes how to write an API program using the REXX run-time interface.
- Chapter 6, “REXX error handling” on page 97 explains how to handle REXX errors.

Complete descriptions of the API commands, as well as some REXX-specific functions and commands, are contained in the *CICSplex SM Application Programming Reference* .

Chapter 2. Using the CICSplex SM API

This chapter introduces a variety of concepts that you need to understand before you use the CICSplex SM API, including managed objects, resource tables, and result sets.

CICSplex SM managed objects

CICSplex SM is an object-oriented system. This means that each resource in the CICSplex SM environment is considered to be an instance of an object. Each object is considered to be a specific type and each has a unique, formally defined name.

Types of managed objects

There are various types of objects in the CICSplex SM environment. Some objects, such as CICS systems, programs, and transactions are real-world resources that CICSplex SM manages. Definition objects, such as monitor specifications and workload definitions, are resources created solely for use within CICSplex SM. An event is an example of a run-time object that is generated as a result of CICSplex SM processing.

The CICSplex SM managed objects can be grouped into the following categories:

- Managed CICS resources
 - CICS resources
 - Monitored CICS resources
- CICS resource definitions
- CICSplex SM definitions
- CICSplex SM manager resources
- CICSplex SM notifications
- CICSplex SM meta-data.

Managed CICS resources

These objects represent actual CICS resources that exist in the CICS systems being managed by CICSplex SM. Each object of this type describes a CICS resource that CICSplex SM can report on and manipulate. Managed objects exist for all the resources that are available to CICSplex SM using standard CICS interfaces. In some cases, the CICSplex SM managed objects offer a more definitive representation of the resources than CICS does. For example, the LOCTRAN and REMTRAN objects, which CICSplex SM uses to separately represent local transactions and remote transactions, are combined by CICS as transactions.

In addition to the standard CICS resources, CICSplex SM creates managed objects as a result of its resource monitoring activity. Monitored CICS resources contain a subset of the resource attributes, normally those that reflect the state and consumption characteristics of the resource. In addition, CICSplex SM may provide derived attributes that show resource utilization as an average, rate, or percentage. MLOCTRAN and MREMTRAN are examples of monitored CICS resource objects;

they are derived from the LOCTRAN and REMTRAN CICS resource objects. A monitored CICS resource object can exist after the associated CICS resource object is removed from the CICS system, or even after the system itself is shut down.

CICS resource definitions

These objects represent definitions of CICS resources that CICSplex SM can assign to, and possibly install in, CICS systems. The actual definitions are stored in the CICSplex SM data repository as definition records. For example, the TRANDEF object represents a CICS transaction that can be assigned both locally and remotely to multiple CICS systems throughout the CICSplex.

Assigning CICS resources to CICS systems enables CICSplex SM to manage those resources as a logical group, such as an application. In addition, CICSplex SM can actually install instances of a resource in CICS systems that support the EXEC CICS CREATE command.

CICSplex SM definitions

These objects represent the definitions that are used by CICSplex SM management applications. The actual definitions are stored in the CICSplex SM data repository as definition records. For example, the MONSPEC object represents a user-defined monitor specification that CICSplex SM uses to establish resource monitoring in a CICS system.

Any changes you make to CICSplex SM definitions are automatically distributed throughout the CICSplex. In addition, certain definitions are bound to other definitions for the purpose of referential integrity. If you remove one of these definitions, all the related definitions are also removed. For example, removing a CPLEXDEF object causes all definition objects for that CICSplex to be automatically removed from all CMASs that manage the CICSplex.

CICSplex SM manager resources

These objects represent run-time resources that are either built from CICSplex SM definitions or created by CICSplex SM management applications during processing. You can manipulate a CICSplex SM manager resource without necessarily affecting the underlying definition. The RTAACTV object is an example of a CICSplex SM manager resource; it describes the currently installed RTADEF and STATDEF definition objects.

There are other CICSplex SM manager resources that are not directly related to any definition. For example, the CRESCONN object is a Topology Services resource map that describes the CICS connections in an active MAS.

CICSplex SM notifications

CICSplex SM notifications are really messages that are generated asynchronously by a CICSplex SM managed object. Notifications describe an interesting event related to the object. CICSplex SM manager resources can register interest in one or more of these events. When a notification is generated, the manager resource performs whatever processing is needed based on the event that occurred.

An API program can also register interest in events that generate CICSplex SM notifications. The EMSTATUS, EMASSICK, and EMASWELL objects are examples of notification messages generated by the CICSplex SM MAS agent. These notifications describe the current state of the MAS.

The ERMxxxx objects are generated by CICSplex SM when a Topology resource map is changed. CICSplex SM maintains resource maps which describe the topology of certain CICS resources in the MASs. CICS resources for which resource maps are maintained have a corresponding ERMxxxx notification object. The CICSplex SM agent detects the installs and discards of these CICS resources and causes the Topology resource map to be updated. For example, if a file definition is installed in a MAS, the Topology resource map will be changed and an ERMCFILE notification will be generated. The ACTION attribute of the ERMCFILE notification indicates that an install has occurred. Furthermore, for CICS/MVS, CICS/ESA 3.3, and CICS/VSE MASs, the CICSplex SM MAS agent detects updates to these CICS resources. For example, if a program is disabled, the ERMCPRGM notification will be generated with the ACTION attribute indicating an update.

CICSplex SM meta-data

These objects describe the structure of CICSplex SM managed resources. This information is maintained in an object directory that exists in each active CMAS.

An API program can request the following types of meta-data from the object directory:

OBJECT	General characteristics of an object
OBJACT	Valid actions for an object
METADESC	Basic description of an object's attributes
ATTR	Complete description of an object's attributes
ATTRAVA	Valid EYUDA or CVDA values for an attribute

CICSplex SM resource tables

Each CICSplex SM managed object is represented externally by a resource table. A resource table defines all the attributes of an object. The attributes represent the collection of data that is available for that object.

The formal object name is used as the name of the resource table that describes the object's attributes. You identify an object in your API program by specifying its resource table name. For example, to find out about the programs in one or more CICS systems, you could access the PROGRAM object. PROGRAM is the name of the CICSplex SM resource table that describes CICS programs.

Each instance of an object is formatted as a resource table record that describes an actual resource in the CICSplex SM environment. The object attributes are presented in the individual fields of a resource table record. It is important to note that a resource table is not itself an object. A resource table record is merely the format in which information about a managed object is returned by CICSplex SM. This information includes the current attribute values, the actions that the object supports, and the releases of CICS for which the object is valid.

There is a resource table type for each type of CICSplex SM managed object:

Resource table type	Object type
CICS Definition	CICS resource definitions
CICS Resource	CICS resources
CICS Monitored	Monitored CICS resources
CPSM Definition	CICSplex SM definitions

CPSM Manager	CICSplex SM manager resources
CPSM Notification	CICSplex SM notifications
CPSM MetaData	CICSplex SM meta-data
CPSM Configuration	CICSplex SM configuration definitions

For a summary of the CICSplex SM resource tables by type and complete descriptions of specific resource tables see the Resource table summary in the *CICSplex SM Resource Tables Reference*.

Restricted Resource Table Attributes

Certain attributes in the CICSplex SM resource tables are for internal use only; they cannot be modified or manipulated by an API program.

In CICS Resource and CICS Monitored tables, CICSplex SM uses the following attributes to uniquely identify which CICS system contains the resource:

- EYU_CICSNAME
- EYU_CICSREL.

These attributes are included in every CICS Resource and CICS Monitored resource table record. You can specify these attributes in a GROUP command to summarize the records in a result set. However, you should not specify these attributes in an ORDER, SPECIFY FILTER, or SPECIFY VIEW command.

CPSM Definition and CICS Definition tables include a CHANGETIME attribute, which reflects the date and time at which the definition was last modified. CICS Definition tables also include a CREATETIME attribute, which is the date and time at which the definition was created. CICSplex SM is solely responsible for maintaining the CHANGETIME and CREATETIME attributes; you should not attempt to modify these attribute values.

Building a customized resource table record

Normally, when you create a result set, each resource table record contains the complete set of attributes in the format defined by CICSplex SM. There may be times, however, when you want to work with a subset of those attributes or work with them in a different order. The SPECIFY VIEW command lets you decide which attributes of a resource table to include in a record and what order to present them in. In effect, you are building a temporary, custom-made resource table.

You can build views only for resource tables with a type of CICS Resource; you cannot build views for any other type of resource table. Also, a view can be built from the attributes of only one resource table at a time. You cannot combine attributes from different resource tables into a single view.

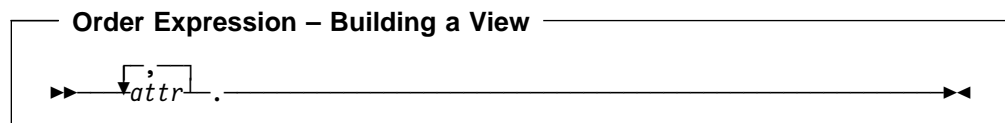
When you build a resource table view, you have to give it a name. The name you assign to a view takes precedence over any existing resource table names. This means you can redefine an existing resource table name to represent a subset of the attributes in a different order than they appear in the original table.

For ease of maintenance of your programs, you are recommended to give unique names to your customized resource table views. If you do not use unique names, you should be aware that you cannot access another view with the same name in the same processing thread, without the programming overhead of discarding the original view. You should also check, when you upgrade your version of

CICSplex SM, that any new resource tables do not duplicate your customized view names.

To tell CICSplex SM which resource table attributes you want to include and in what order, you specify an order expression on the FIELDS option of the SPECIFY VIEW command. This expression is similar to the one you use when sorting records in a result set with the ORDER command. The order expression consists of a list of the attributes to be included in the view.

The syntax of an order expression for building a view is:



where:

attr Is the name of an attribute in the resource table.

You can specify as many attribute names as you like, but the total length of an order expression, including commas and blank spaces, must not exceed 255 characters.

Notes:

1. You cannot specify the EYU_CICSNAME or EYU_CICSREL attributes in an order expression. To identify the CICS system from which a view record was collected, retrieve the OBJSTAT information along with the data. You can do this by specifying the BOTH option on the FETCH command.
2. An order expression must be followed by either blank spaces or null characters to the end of the specified buffer. That is, the buffer length you specify (using the LENGTH option) should not include any data other than an order expression.

For example, to build a limited view of the LOCTRAN resource table, you could specify:

```
TRANID,STATUS,USECOUNT,PROGRAM,PRIORITY,TRANCLASS.
```

Once a view is built, you can specify it on the OBJECT option of a GET command, just as you would the resource table itself. The resource table records returned by GET include only those attributes you named in the order expression on the SPECIFY VIEW command.

Any views that you build are associated with the specific processing thread on which you build them; they cannot be shared by other processing threads. When you terminate your processing thread, any views you built on it are discarded. You can also choose to discard a view at any time by using the DISCARD command.

How to create copybooks for customized resource table records

You can build a structure for your customized view by using the SPECIFY VIEW, GET and FETCH commands to move the data into your structure. For example:

```

*****
*                SPECIFY VIEW                *
*****
STRING 'POOLNAME,MINITEMLEN,QUELENGTH,NUMITEMS,'
'RECOVSTATUS,MAXITEMLEN,LASTUSEDINT,'
'NAME,TRANSID,LOCATION.'
DELIMITED BY SIZE INTO BUFFERA.
MOVE 96 TO BUFFERL.
EXEC CPSM SPECIFY
        VIEW('VTSQSHR')
        FIELDS(BUFFERA)
        LENGTH(BUFFERL)
        OBJECT('TSQSHR')
        THREAD(TTKN(1))
        RESPONSE(SMRESP)
        REASON(SMRESP2)
END-EXEC.

```

Figure 2. SPECIFY VIEW command to build a structure

The associated structure will consist of each attribute specified in the SPECIFY VIEW FIELDS keyword and is shown in Figure 3.

```

01 VTSQSHR.
* Shared Temporary Storage Queue
  02 POOLNAME          PIC X(0008).
* TS Pool Name
  02 MINITEMLEN       PIC S9(0004) USAGE BINARY.
* Smallest item Length in bytes
  02 QUELENGTH        PIC S9(0008) USAGE BINARY.
* Total length in bytes . FLENGT
  02 NUMITEMS         PIC S9(0004) USAGE BINARY.
* Number items in queue
  02 RECOVSTATUS      PIC S9(0008) USAGE BINARY.
* Recovery Status
  02 MAXITEMLEN       PIC S9(0004) USAGE BINARY.
* Largest item length in bytes
* -----*
*   LASTUSEDINT                      *
*   Invalid CICSREL = E330 E410 0300 0311 N500 *
* -----*
  02 LASTUSEDINT      PIC S9(0008) USAGE BINARY.
* Interval since last use
  02 NAME-R           PIC X(0016).
* Queue Name          -- RESERVED WORD --
* -----*
*   TRANSID                      *
*   Invalid CICSREL = E330 E410 0300 0311 N500 *
* -----*
  02 TRANSID          PIC X(0004).
* Trans that created tsqueue
  02 LOCATION         PIC S9(0008) USAGE BINARY.
* Queue Location

```

Figure 3. Structure of a customized view

Note that the EYU-CICSNAME, EYU-CICSREL, and EYU-RESERVED attributes or any field alignment or padding attributes are not used in this structure.

Selecting managed objects

Any given API program is likely to be interested in only a subset of the CICSplex SM managed objects. You can identify the managed objects you want to work with by:

- Setting the context and scope for your program
- Using filter expressions on individual commands.

Setting the context and scope

The set of managed objects that your API program can work with is determined primarily by the context and scope associated with the processing thread. As with all CICSplex SM operations, the context and scope of an API program identify the CICS systems on which the program can act.

In general, you can set the context and scope values as follows:

CONTEXT For most operations in a CICSplex, the context is the name of the CICSplex. For operations related to CMAS configuration (such as defining CICSplexes or CMAS communication links), the context must be a CMAS name.

SCOPE When the context is a CICSplex, the scope can be:

- 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, the scope value is ignored.

You can set a default context and scope for your program by using one of these commands:

CONNECT Defines a default context and scope when the API processing thread is established.

QUALIFY Changes the default context and scope for subsequent commands issued on the thread.

The values you set on either of these commands are in effect for all API commands that use context and scope.

Alternatively, you can specify context and scope values for individual API commands. The following commands support one or both of the **CONTEXT** and **SCOPE** options:

- CREATE
- GET
- LISTEN
- PERFORM OBJECT
- REMOVE
- UPDATE.

The context and scope values you set on any of these commands are in effect for that command alone. If you specified a default context and scope for the thread, the values on any of these commands temporarily override the default values. If you did not specify a default context and scope and you issue a command that

selecting managed objects

expects these values (such as GET), you must specify a context and scope on the command.

Using filter expressions

A request for CICSplex SM managed object data can produce a large number of resource table records. The default is to return all the resource table records that exist for a given object within the current context and scope. For example, if you ask for PROGRAM object data, you receive a resource table record for every program in every CICS system in the current context and scope. However, if you are only interested in certain programs, you can use a filter expression to limit the number of records returned based on the current values of certain PROGRAM attributes.

How you can use filter expressions

You can use filter expressions in one of two ways:

- With the CRITERIA option of a GET or PERFORM OBJECT command to filter the resource table records returned by that command. The filter expression is used only once and is discarded when the command that used it completes its processing.
- With a SPECIFY FILTER command to define a filter that can be used repeatedly.

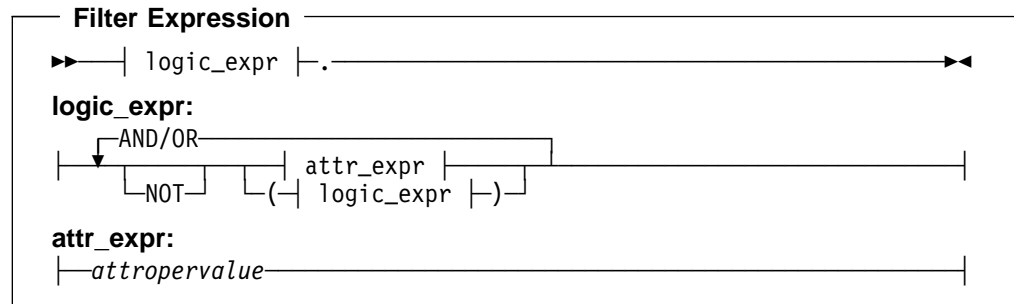
Once a filter is defined, you can use it with these commands to limit the resource table records being processed:

- COPY
- DELETE
- FETCH
- GET
- GROUP
- LISTEN
- LOCATE
- MARK
- PERFORM OBJECT
- PERFORM SET
- REFRESH
- SET
- UNMARK

A filter expression that you define with the SPECIFY FILTER command is available to your program until you either discard it (with the DISCARD command) or terminate the processing thread.

How to build a filter expression

A filter expression is a character string that defines logical expressions to be used in filtering resource table records. A filter expression can be made up of one or more attribute expressions in the form:



where:

attr Is the name of an attribute in the resource table.

You can name the same attribute more than once in a filter expression.

Note: You cannot specify the EYU_CICSNAME or EYU_CICSREL attributes in a filter expression.

oper Is one of the following comparison operators:

- < Less than
- <= Less than or equal to
- = Equal to
- >= Greater than or equal to
- > Greater than
- ≠ Not equal to

value Is the value for which the attribute is being tested. The value must be a valid one for the resource table attribute.

Generic values

If the attribute accepts character data, this value can be a generic. Generic values can contain:

- An asterisk (*), to represent any number of characters, including zero. The asterisk must be the last or only character in the specified value. For example:

TRANID=PAY*

- A plus sign (+), to represent a single character. A + can appear in one or more positions in the specified value. For example:

TRANID=PAY++96

Imbedded blanks or special characters

If the value contains imbedded blanks or special characters (such as periods, commas, or equal signs), the entire value string must be enclosed in single quotes. For example:

TERMID='Z AB'

To include a single quote or apostrophe in a value, you must repeat the character, like this:

```
DESCRIPTION='October' 's Payroll'
```

Note: Be sure to consider the quoting conventions of your programming language when using single quotes in a CICSplex SM value string.

Hexadecimal data

If the attribute has a datatype of HEX the value must be in hexadecimal notation.

For example, the NAME attribute of the REQID resource table is a HEX datatype. To specify a name equal to 01234567 the value, using hexadecimal notation, would be

```
NAME=F0F1F2F3F4F5F6F7
```

AND/OR Combines attribute expressions into compound logic expressions using the logical operators AND and OR, like this:

```
attr_expr AND attr_expr.
```

Filter expressions are evaluated from left to right. You can use parentheses to vary the meaning of a filter expression. For example, this expression:

```
attr_expr AND (attr_expr OR attr_expr).
```

has a different meaning than this one:

```
(attr_expr AND attr_expr) OR attr_expr.
```

NOT Negates one or more attribute expressions.

You can negate a single attribute expression, like this:

```
NOT attr_expr
```

You can also negate multiple attribute expressions or even a whole filter expression, like this:

```
NOT (attr_expr OR attr_expr).
```

Note that you must place parentheses around the attribute expressions (or the filter expression) to be negated.

Note: A filter expression must be followed by either blank spaces or null characters to the end of the specified buffer. That is, the buffer length you specify (using the LENGTH option) should not include any data other than a filter expression.

For example, the following is a simple filter expression that you could use to select LOCTRAN objects representing local transactions that are enabled and have a storage violation count greater than zero:

```
STATUS=ENABLED AND STGVCNT>0.
```

You can build more complex filter expressions to select objects with a very specific combination of attributes. For example, to select LOCTRAN objects that:

- Have a transaction ID starting with P
- Have a program name starting with PAY

- Are enabled
- Have a nonzero use count and storage violations, or have been restarted.

you could specify a filter expression like this:

```
(TRANID=P* AND PROGRAM=PAY* AND STATUS=ENABLED) AND
((USECOUNT>0 AND STGVCNT>0) OR NOT RESTARTCNT=0).
```

Note that the RESTARTCNT attribute in this example could also have been specified with the greater than operator instead of the NOT operator.

Working with result sets

CICSplex SM places the resource table records that you select in a result set. A result set is a logical group of resource table records that can be accessed, reviewed, and manipulated by an API program.

A result set can be created in one of two ways:

- By a direct API request to obtain resource data. The GET command is the primary means of collecting resource data and creating a result set.
- By an API request that manipulates one result set to create another. COPY is an example of a command that can create a new result set from the records in an existing one. The result set from which records are being copied is referred to as the source result set. The one being copied to is the target result set.

The resource table records in a result set must all represent one type of managed object. That is, a result set that contains PROGRAM resource table records cannot also contain LOCTRAN resource table records. The resource table records must also be collected from the same CICSplex SM context. So a result set that contains records from one CICSplex cannot be used to hold records from any other CICSplex. Once a result set is created, its resource type and context are fixed. The only way to change the type or context of a result set is to completely replace the contents of the result set with new resource table records.

Keep in mind that a resource table record in a result set is not the actual managed object; it is a report of the managed object's attributes at the point in time when data was collected. This is an important distinction because the actual managed object may have changed or may no longer exist by the time the resource table record is returned to your program. The number of records returned may vary as managed objects come and go, but the structure of the records in a result set remains constant.

A simple API program might deal with only one result set at a time. Each new request for data could create a result set that replaces the previous one. A more complex program might maintain several result sets concurrently and control the retention of those result sets more directly.

An overview of result set commands

You can use the following commands to create result sets and manage the resources that they represent:

GET Returns a result set containing selected resource table records that represent instances of a managed resource.

working with result sets

- PERFORM** Performs an action on one or more managed resources. PERFORM SET acts upon the resource table records in an existing result set. PERFORM OBJECT does not require a result set to exist; it creates one implicitly.
- REFRESH** Refreshes the data for some or all of the managed resources as represented by resource table records in a result set.
- SET** Modifies the attributes of one or more managed resources as represented by resource table records in a result set.

These commands affect not only the resource table records in a result set, but also the managed resources that those records represent. Figure 4 illustrates the relationship of these commands to the API environment.

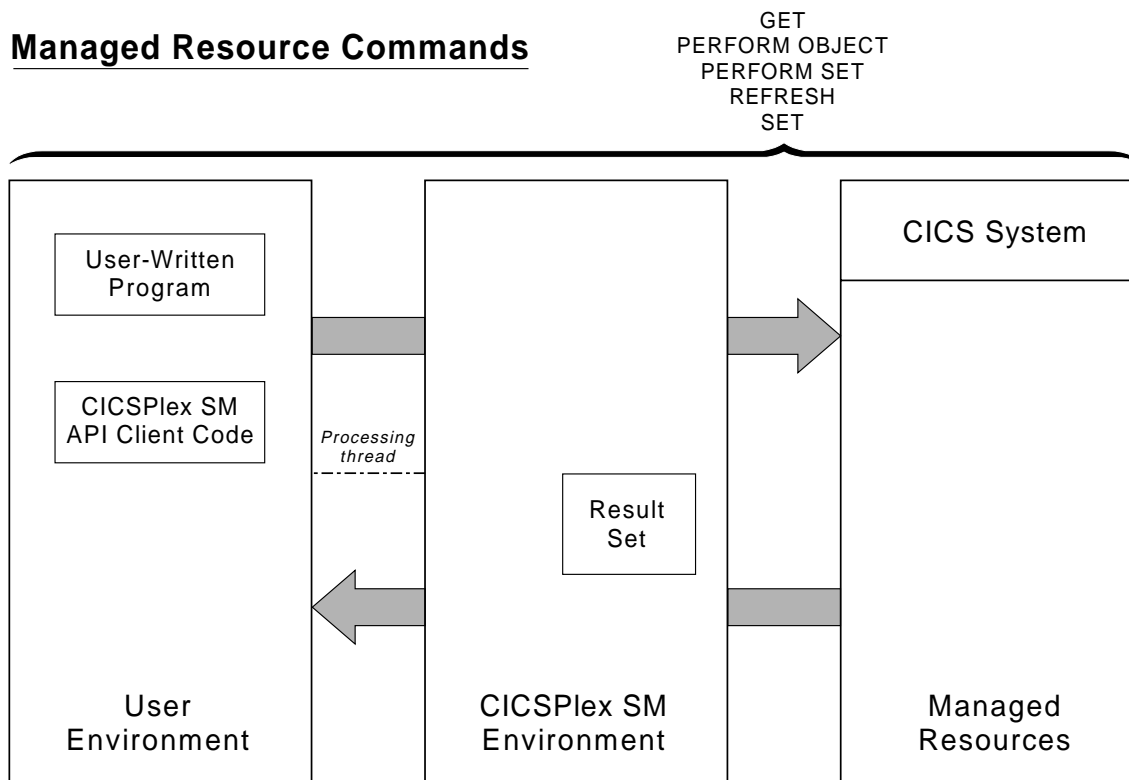


Figure 4. API commands that manipulate managed resources

Once a result set is created, you can perform various operations on the records it contains. You can sort, mark, copy, delete, and summarize the records in a result set. Most importantly, perhaps, you can retrieve records from a result set into local storage where they can be processed by your program.

You can use the following commands to manipulate one or more records in a result set:

- COPY** Copies some or all of the resource table records in one result set to another result set.
- DELETE** Deletes one or more resource table records from a result set.
- FETCH** Retrieves data and status information for one or more resource table records in a result set.

- GROUP** Returns a summarized result set by grouping some or all of the resource table records in a result set.
- LOCATE** Positions the record pointer within a result set.
- MARK** Marks selected resource table records in a result set.
- ORDER** Sorts the resource table records in a result set.
- UNMARK** Removes the marks placed on resource table records by a previous MARK command.

These commands affect only the current contents of a result set; they have no impact on the managed resources that the result set represents. Figure 5 illustrates the relationship of these commands to the API environment.

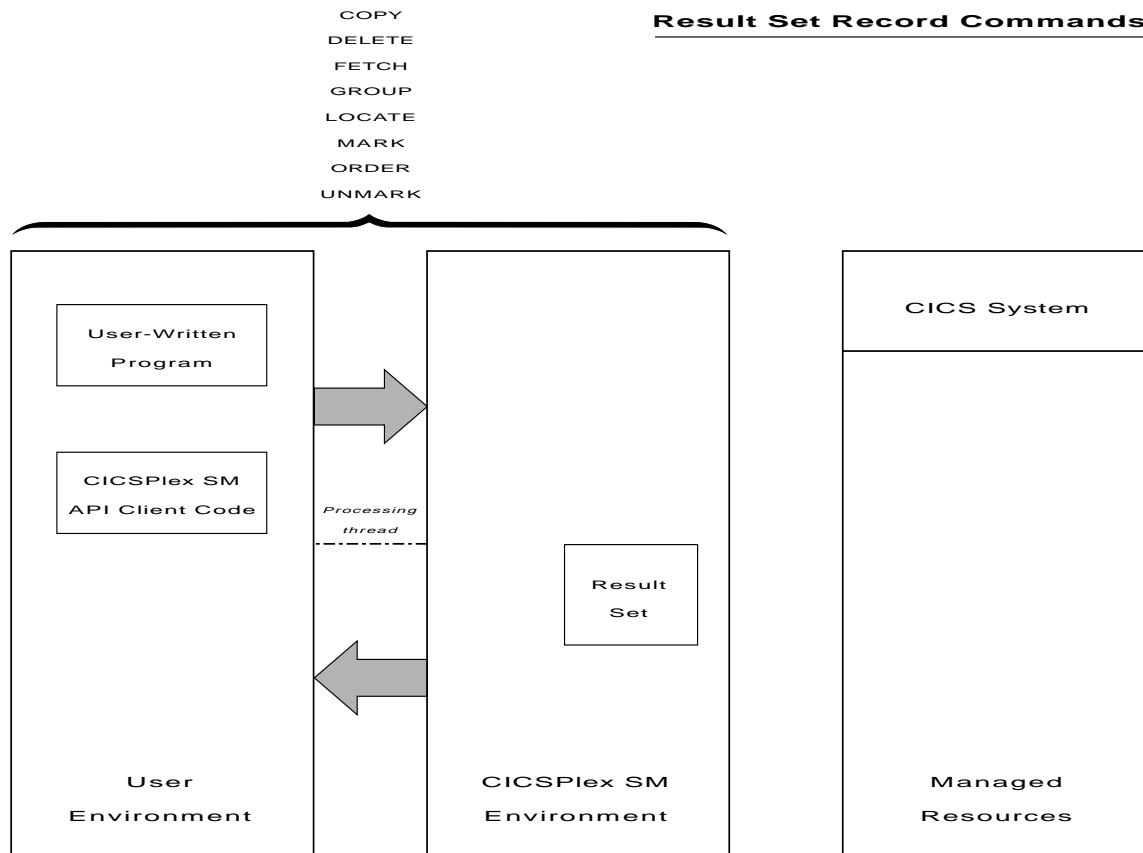


Figure 5. API commands that manipulate result set records

CICSplex SM also provides tools for managing result sets as a whole: filters and views for controlling the contents of a result set and commands for reviewing and discarding result sets.

You can use the following commands to manage result sets and their contents:

- DISCARD** Discards a result set.
- QUERY** Retrieves information about a result set and the resource table records it contains.
- SPECIFY FILTER** Defines an attribute or value filter that can be used to control the contents of a result set.

SPECIFY VIEW Builds a customized view of a resource table that can be used to control the contents of a result set

These commands affect only an existing or newly created result set; they have no impact on the managed resources that the result set represents. Figure 6 illustrates the relationship of these commands to the API environment.

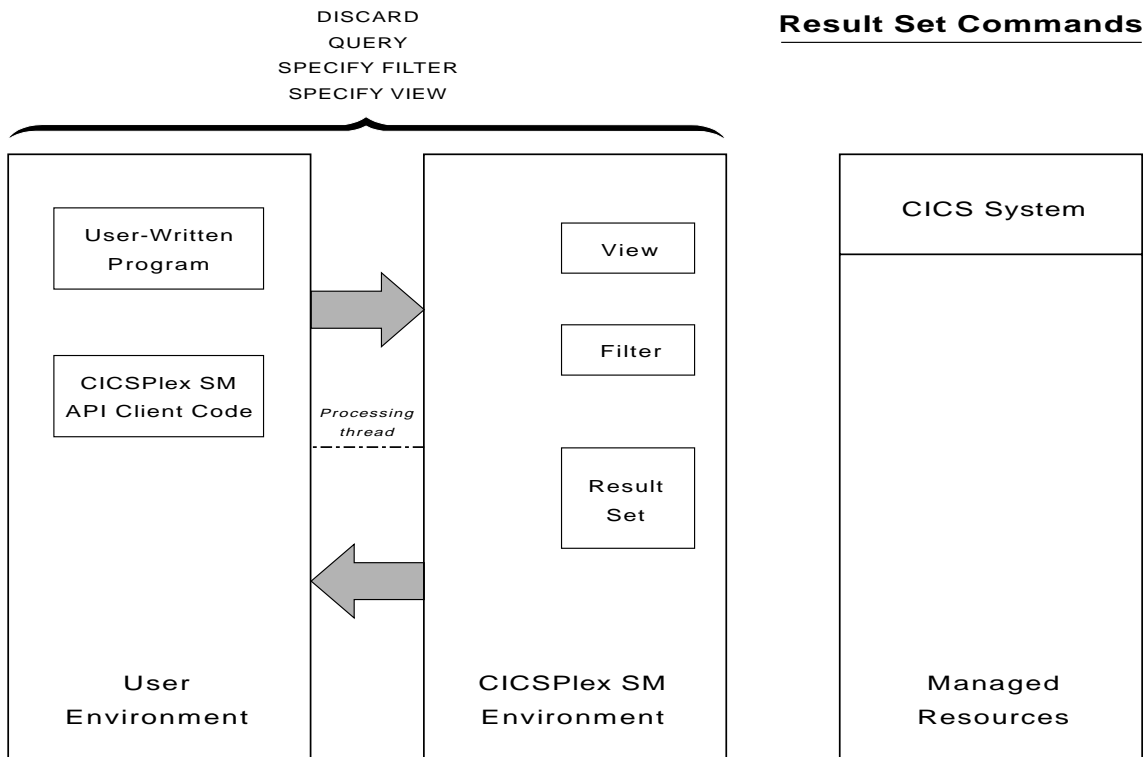


Figure 6. API commands that manipulate result sets

Retrieving records from a result set

Once you have created a result set (using the GET command), you can transfer some or all of the records it contains to local storage for processing. You can use the FETCH command to retrieve a single resource table record, multiple selected records, or the entire result set at one time.

Each resource table record that you retrieve contains current data about the managed resource that it represents. Each record also contains certain status information that is maintained by CICSplex SM. This status information, which is presented in the form of a resource table called OBJSTAT, includes:

- The name and release level of the CICS system from which data was collected
- The context in effect when data was collected
- The name and type of the managed resource
- The last operation performed against the resource
- The current status of the resource.

In effect, each record in a result set contains a pair of resource tables: an instance of the OBJSTAT resource table followed by an instance of the resource table that was requested. The managed resource data and the OBJSTAT status information can be retrieved either as a pair or separately, depending on the option you specify with the FETCH command:

- DATA** Retrieves only the specified resource table data.
- STATUS** Retrieves only the OBJSTAT status information.
- BOTH** Retrieves both the resource table data and the OBJSTAT status information.

Figure 7 illustrates the information available in result set records and the FETCH commands you can use to retrieve that information.

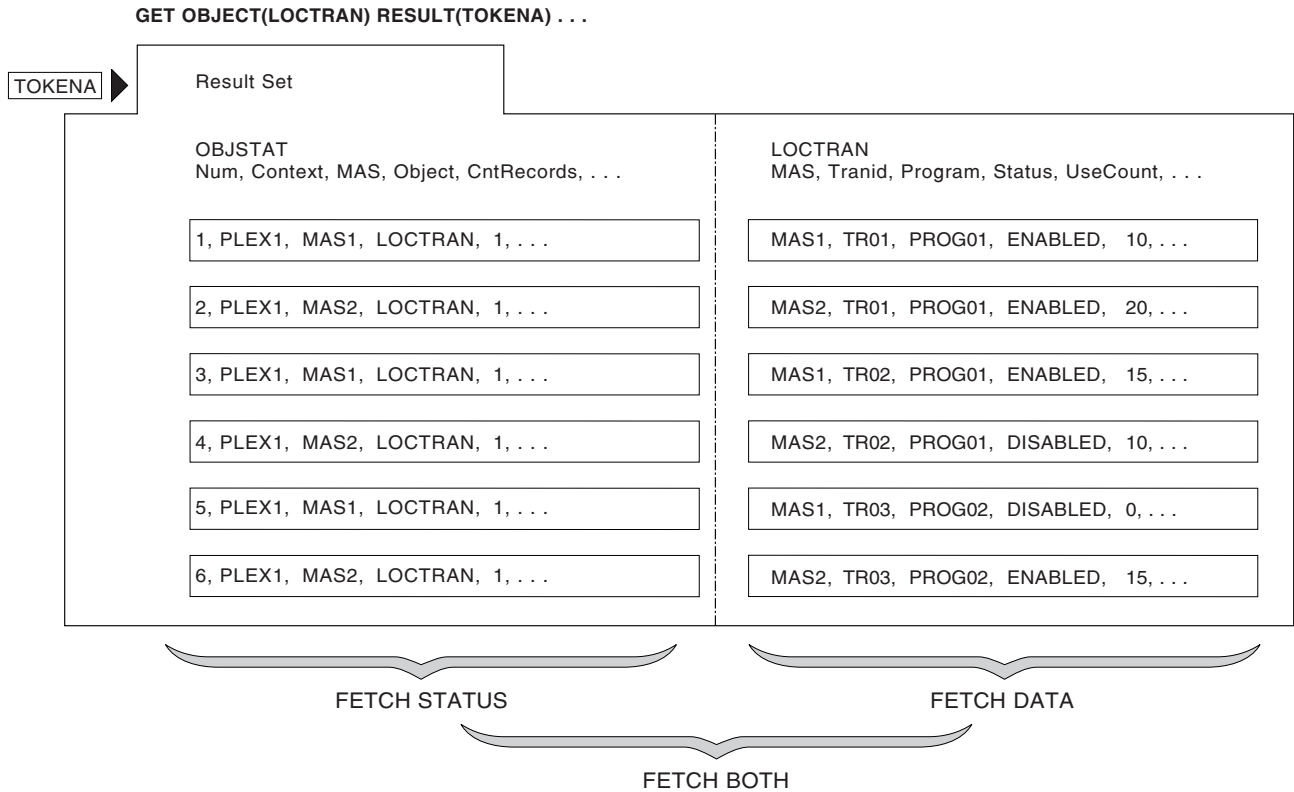


Figure 7. Using FETCH to retrieve result set records

The result set referenced by TOKENA was created by issuing a GET command for LOCTRAN records. Each record in the result set consists of LOCTRAN data and OBJSTAT data.

You can use the FETCH commands shown in Figure 7 to selectively retrieve some or all of the data. For example, Figure 8 shows the output of a FETCH DATA command.

FETCH DATA ALL RESULT(TOKENA) INTO(AREA1) ...

MAS1, TR01, PROG01, ENABLED, 10, ...
MAS2, TR01, PROG01, ENABLED, 20, ...
MAS1, TR02, PROG01, ENABLED, 15, ...
MAS2, TR02, PROG01, DISABLED, 10, ...
MAS1, TR03, PROG02, DISABLED, 0, ...
MAS2, TR03, PROG02, ENABLED, 15, ...

Figure 8. Sample FETCH DATA output

Positioning the record pointer in a result set

CICSplex SM maintains a current record pointer in each result set. When you first create a result set (with a GET command, for example), the pointer is positioned at the top of the result set. The first command that you issue against the result set affects the first record.

In most cases, when you issue FETCH commands to retrieve records from the result set, the record pointer is positioned to the next record in the result set (that is, the record following the last record that was fetched). However, certain API commands always act upon the last record that was fetched. When you issue any of these commands after a FETCH command, the record pointer is not advanced to the next record:

- COPY
- DELETE
- MARK.
- UNMARK.

The record pointer in a result set may actually move either forward or backward, depending on the direction in which you are retrieving records. If you issue a FETCH command and no records are found that match the specified criteria, no records are retrieved. In that case, the pointer is positioned to the top or bottom of the result set, depending on the direction the pointer was moving.

The GET and FETCH commands leave the record pointer in specific, predefined positions, but other API commands do not. Many API commands manipulate records or update the data in a result set. The position of the record pointer after one of these commands depends on a combination of factors, including the options that you specified on the command. The pointer may have moved forward or backward one or more records, or it may be positioned to the top or bottom of the result set. If you specified the CURRENT option, the record pointer does not move; it remains positioned on the current record after the command is complete.

For this reason, CICSplex SM provides the LOCATE command, which lets you explicitly position the record pointer within a result set. If you want to use the record pointer after issuing any of these commands, first use the LOCATE command to reposition it:

- COPY
- DELETE
- GETDEF
- GROUP
- MARK
- ORDER
- PERFORM OBJECT
- PERFORM SET
- REFRESH
- SET
- UNMARK.

Processing selected records in a result set

If you want to process a subset of the resource table records in a result set, you can identify the records you are interested in by:

- Using the SPECIFY FILTER command to define a filter for selecting records, as described in “Using filter expressions” on page 20.
- Using the MARK and UNMARK commands to mark the records.

Using MARK and UNMARK

The MARK command enables you to mark some or all of the resource table records in a result set for future reference. The UNMARK command removes existing marks from selected records. Once you have marked records in a result set, you can refer to the records that are either marked or not marked in subsequent commands. The following API commands support the MARKED and NOTMARKED options:

- COPY
- DELETE
- FETCH
- GROUP
- LOCATE
- PERFORM SET
- REFRESH
- SET

For example, Figure 9 on page 30 shows a result set in which selected resource table records have been marked. The MARKED option is then used with the FETCH command to retrieve only those records that are marked.

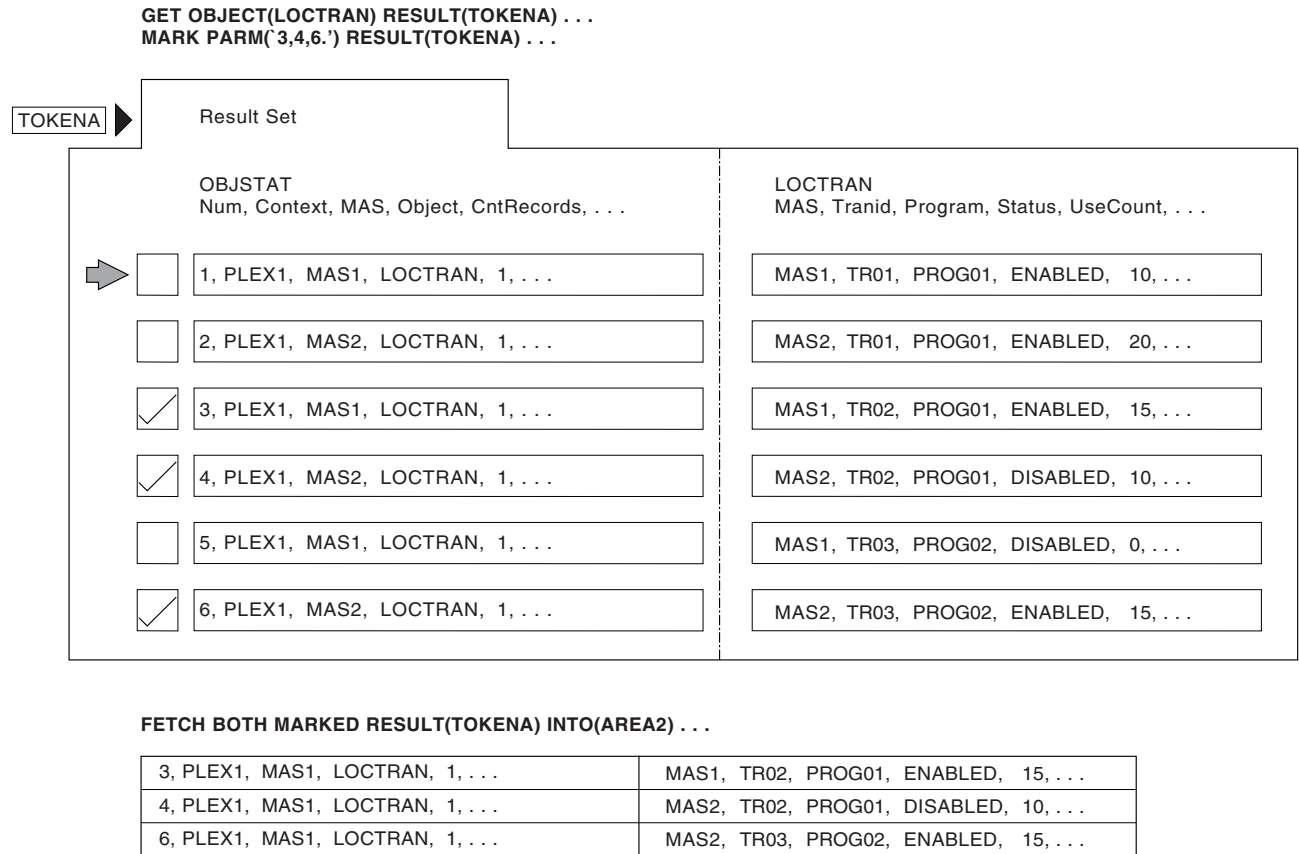


Figure 9. Marking and retrieving records in a result set

Identifying the records to be marked

By default, when you issue a MARK or UNMARK command, only the current resource table record is marked or unmarked. But there are a variety of ways that you can identify the records to be marked:

- To mark a specific record other than the current record, use the POSITION option and identify the record by its relative position in the result set.
- To mark one or more records that meet previously defined filtering criteria, use the FILTER or NOTFILTER option.
- To mark all the records in a result set, use the ALL option.

In addition to these options, you can use the PARM option to identify a list of records to be marked. To use the PARM option, you specify a character string of record numbers in a parameter expression. The parameter expression can contain:

- Individual record numbers, separated by commas.
- Ranges of record numbers, with the low and high numbers separated by a colon.

The whole parameter expression must end with a period.

For example, to mark records 1, 3, 6 through 9, and 24 in a result set, you would specify:

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

When you use the PARM option, you must also use the PARMLEN option to specify the length of the buffer that contains the parameter expression.

Notes:

1. Negative values and 0 are not valid record numbers. If you specify an invalid record number, the MARK (or UNMARK) command returns RESPONSE and REASON values of INVALIDPARM PARM.
2. If you mistakenly specify the higher value in a range first (such as 9:6), CICSPlex SM reverses the values to produce a valid range.
3. If you mistakenly specify a single value preceded or followed by a colon (such as 6:), the colon is ignored. CICSPlex SM marks only the specified record.

Identifying records that could not be marked

When you are marking or unmarking records, it might be useful to know if all the records you identified were successfully processed. For example, you might mistakenly ask CICSPlex SM to mark or unmark a record that was previously deleted from the result set. Or you might identify a record number that is out of range for the result set.

You can use the COUNT option on a MARK or UNMARK command to determine the number of records that could not be marked or unmarked. You can also use the INTO and LENGTH options to identify a buffer to receive a list of records that could not be marked. When deciding on the length of the INTO buffer, keep in mind that it 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). Furthermore, all record numbers are listed individually (not by range) in the INTO buffer and are separated by commas. So if you specified the PARM option like this:

```
PARM('1,3:6,12,15.')
```

the INTO buffer would have to be long enough to hold the following character string:

```
1,3,4,5,6,12,15
```

If the INTO buffer you specify is not long enough to hold a complete list of records that could not be marked, you receive a RESPONSE value of WARNING AREATOOSMALL. In that case, the INTO buffer returns a partial list of records and the LENGTH value is set to the buffer length that would be required for a complete list. You could then resubmit the MARK command with the appropriate LENGTH value to determine which records could not be marked.

How to remove the marks in a result set

You can use the UNMARK command to remove some or all of the marks placed on resource table records by a previous MARK command. However, if you want to mark other records at the same time, you can save a step by using the RESET option of the MARK command.

By default, the records you specify on a MARK command are marked in addition to any records that are already marked in the result set. That is, any resource table records that were marked previously remain marked unless you use the RESET option. RESET wipes the result set clean of any previous marks. So the records identified on the current MARK command are the only records marked when

processing is complete. Using the RESET option on a MARK command is an alternative to using the UNMARK command before the MARK command.

Note: Any marks that you placed on resource table records are also removed when you use the COPY command to copy those records from one result set to another.

Summarizing the records in a result set

If you want to analyze or modify a large number of records in a result set, you might find it useful to summarize those records. The GROUP command lets you summarize the records in a result set based upon the value of some resource table attribute.

When you issue a GROUP command, CICSPlex SM summarizes the records in one result set to create a new, summarized result set. A summarized result set is a special type of result set. It contains summary resource table records that correspond to one or more records in the source result set.

For example, you could use the GROUP command to summarize a result set that contains LOCTRAN resource table records. If you want to group the records according to the value of the STATUS attribute, the summarized result set would contain, at most, two records: one representing those records with a STATUS value of ENABLED, and one representing those with a STATUS of DISABLED. Figure 10 illustrates this use of the GROUP command.

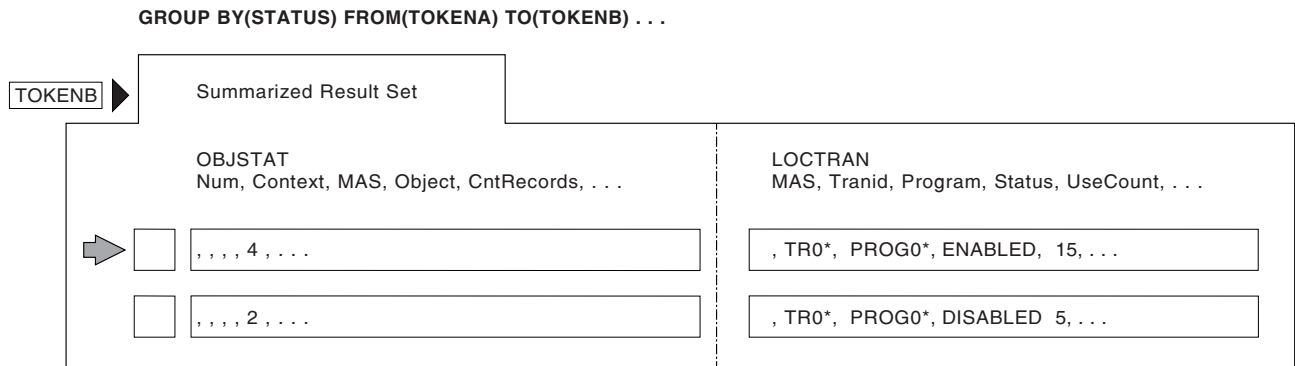


Figure 10. Using GROUP to summarize result set records

In general, you can work with a summarized result set in the same ways that you do a regular result set. You can use the FETCH command to retrieve records from a summarized result set. You can also retrieve the individual records of the source result set on which the summary is based. The DETAIL option of the FETCH command lets you retrieve that subset of records in the source result set that correspond to a particular summary record.

Figure 11 on page 33 shows an example of fetching the detail records associated with a summary record. In this case, the summary record was a LOCTRAN record that represented all enabled transactions.

FETCH DETAIL RESULT (TOKENB) INTO (AREA3) . . .

MAS1, TR01, PROG01, ENABLED, 10, . . .
MAS2, TR01, PROG01, ENABLED, 20, . . .
MAS1, TR02, PROG01, ENABLED, 15, . . .
MAS2, TR03, PROG02, ENABLED, 15, . . .

Figure 11. Sample FETCH DETAIL output

You can modify the records in a summarized result set using the PERFORM or SET commands. This is equivalent to modifying all the records in the source result set that are represented by a given summary record. However, since each record in a summarized result set has a single OBJSTAT record associated with it (rather than one for each of the source records being modified), you may want to use the FETCH DETAIL command to determine the results of a summary action.

The OBJSTAT records in a source result set are not summarized when you issue a GROUP command. So the OBJSTAT records in a summarized result set do not represent the OBJSTAT information for all of the source records. However, the OBJSTAT records in a summarized result set do include a summary count, which indicates how many source records were combined to produce each summary record.

A summarized result set and its source result set should be thought of as a pair to be used together. They share certain attributes and the summarized result set has certain dependencies on the source result set:

- A summarized result set cannot exist without the source result set from which it was built. If you discard a source result set, all the summarized result sets that were built from it are also discarded.
- You can reuse a summarized result only to resummairize the records in the same source result set. An existing summarized result set cannot be used as the target of a GROUP command for a different source result set.
- If you modify a source or summarized result set in any way, all the summarized result sets that have been built from the source result set are rebuilt.

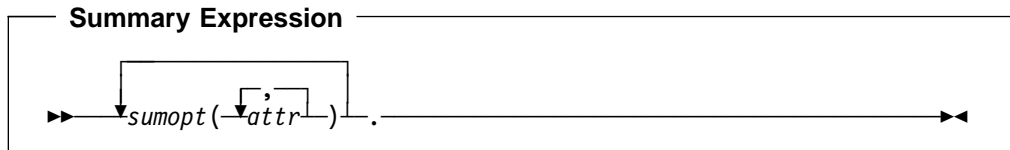
Note: To prevent this from happening, you can specify the NOREFRESH option on the PERFORM or SET command.

Specifying summary expressions

The attributes of a summary record are set according to a summary option that is appropriate for the attribute's data type. For each resource table attribute, CICSplex SM defines a default summary option. CICSplex SM uses these defaults when summarizing records unless you explicitly override them.

You tell CICSplex SM how to summarize the attributes in a record by specifying a summary expression on the SUMOPT option of the GROUP command. A summary expression is a character string that consists of one or more summary options and the resource table attributes to which they apply.

The syntax of a summary expression is:



where:

- sumopt** Is the summary option to be used for the specified resource table attributes:
- AVG** Provides the average attribute value. Valid for numeric fields only.
 - DIF** Provides those characters that are common to all underlying records and displays an asterisk (*) for those not common. Valid for character fields only.
 - LIKE** Provides the CVDA or EYUDA value, if all records contain a common value. Otherwise, displays N/A. Valid for CVDA and EYUDA fields only.
 - MAX** Provides the maximum attribute value. Valid for numeric fields only.
 - MIN** Provides the minimum attribute value. Valid for numeric fields only.
 - SUM** Provides the sum of the attribute values. Valid for numeric fields only.

You can specify the same summary option more than once in a summary expression.

- attr** Is the name of an attribute in the resource table.
- You can specify as many attribute names for each summary option as you like.

Note: A summary expression must be followed by either blank spaces or null characters to the end of the specified buffer. That is, the buffer length you specify (using the LENGTH option) should not include any data other than a summary expression.

For example, you could use a summary expression like this when grouping LOCTRAN records:

```
SUM(USECOUNT) MAX(PRIORITY, TWASIZE) .
```

By default, the values for these attributes would be averaged. But this summary expression specifies that each summary record should include the sum of all USECOUNT values and the maximum PRIORITY and TWASIZE values.

Sorting the records in a result set

The records in a result set are normally sorted by the key attributes for that resource table. In the case of CICS Resource and CICS Monitored tables, records are sorted by the CICS system from which they were collected. In working with result sets, you may find it easier to process the records if they are in some logical order of your own choosing. The ORDER command lets you sort the records in a result set according to the values of a particular resource table attribute.

You tell CICSplex SM how to sort the records by specifying an order expression on the BY option of the ORDER command. An order expression is a character string that consists of one or more attribute names to be used in sorting the resource table records.

The syntax of an order expression for sorting records is:



where:

attr Is the name of an attribute in the resource table.

You can specify as many attribute names as you like, but the total length of an order expression, including commas and blank spaces, must not exceed 255 characters.

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

/D Indicates the attribute values should be sorted in descending order. By default, the values are sorted in ascending order.

Note: An order expression must be followed by either blank spaces or null characters to the end of the specified buffer. That is, the buffer length you specify (using the LENGTH option) should not include any data other than an order expression.

For example, to sort a result set of LOCTRAN records by transaction ID and enabled status, you could specify:

```
TRANID,STATUS.
```

In this example, transaction ID is the primary sort key and enabled status is the secondary sort key.

To sort records in descending order of use count, add /D to the end of the attribute name, like this:

```
USECOUNT/D
```

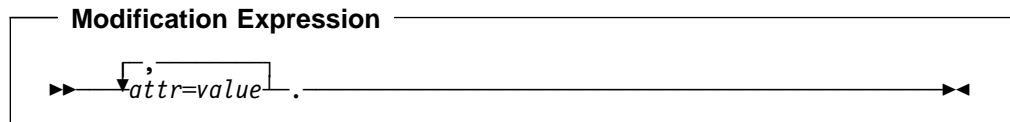
Modifying managed resources

This section describes various ways in which you can modify the resources managed by CICSplex SM. The actions described here are issued against resource table records in a result set. However, the changes that you request are made to the actual resources which those records represent.

Modifying resource attributes

You can change the current value of a resource attribute by using the SET or UPDATE command. SET modifies the attributes of a CICS resource, while UPDATE modifies CICSplex SM and CICS definitions. The MODIFY option of these commands accepts a modification expression, which is a character string that defines the attribute changes to be made.

A modification expression can be made up of one or more attribute expressions in the form:



where:

attr Is the name of a modifiable attribute in the resource table.

value Is the value to which you want the attribute set. The following restrictions apply:

- The value must be a valid one for the attribute.
- If the value contains imbedded blanks or special characters (such as periods, commas, or equal signs), the entire value string must be enclosed in single quotes, like this:

```
DESCRIPTION='Payroll.OCT'
```

- To include a single quote or apostrophe in a value, you must repeat the character, like this:

```
DESCRIPTION='October''s Payroll'
```

Note: Be sure to consider the quoting conventions of your programming language when using single quotes in a CICSplex SM value string.

Note: A modification expression must be followed by either blank spaces or null characters to the end of the specified buffer. That is, the buffer length you specify (using the LENGTH option) should not include any data other than a modification expression.

For example, to disable one or more local transactions (LOCTRAN), you could specify:

```
STATUS=DISABLED.
```

in the MODIFY option of a SET command.

If you issue a SET command against CICS systems that do not support the requested modification, the request is ignored for those CICS systems. If your context and scope consist solely of CICS systems that do not support the modification, you receive RESPONSE and REASON values of NOTAVAILABLE SCOPE.

To change the task storage location of a CICS transaction definition (TRANDEF), you could specify:

```
TASKDATALOC=ANY
```


in the MODIFY option of an UPDATE command.

Note that the MODIFY option of UPDATE is valid only for CICS Definition resource tables.

For a list of the attributes for each resource and their valid values, refer to the *CICSplex SM Resource Tables Reference*.

Performing an action against a resource

In addition to modifying individual attributes, you can also perform actions against many resources by using either of the PERFORM commands, PERFORM OBJECT or PERFORM SET. The difference between these two commands is that PERFORM SET performs an action against the resource table records in an existing result set, while PERFORM OBJECT first creates a result set and then performs the requested action.

Some actions are self-contained and self-explanatory; specifying the action is enough to indicate the changes to be made to the resource. For example, you can discard a local file by issuing the DISCARD action against a LOCFILE resource table record.

Other actions require you to specify additional parameters. For these actions, you have to specify the PARM option on the PERFORM command. The PARM option accepts a parameter expression, which is a character string that defines the parameters required for an action to complete. For example, to disable a local file, you must indicate how to handle a file that is currently busy. To do that, you could specify the following parameter expression:

```
PARM('BUSY(cvda).')
```

where cvda is a valid CVDA value for the file busy condition.

If you issue a PERFORM command against CICS systems that do not support the requested action, the request is ignored for those CICS systems. If your context and scope consist solely of CICS systems that do not support the action, you receive RESPONSE and REASON values of NOTAVAILABLE SCOPE.

For a list of the valid actions for each resource and their required parameters, refer to the *CICSplex SM Resource Tables Reference*.

Working with CICSplex SM and CICS definitions

When you work with CICSplex SM and CICS definitions there are some special API commands and command options available.

Creating, updating, and removing definitions

You can use the following API commands to maintain the CICSplex SM and CICS definitions in your data repository:

- | | |
|---------------|---|
| CREATE | Creates a new CICSplex SM or CICS definition using the attribute values you specify. The new definition is stored in the data repository. |
| UPDATE | Updates an existing CICSplex SM or CICS definition according to the attribute values you specify. The updated definition replaces the existing definition in the data repository. |

REMOVE Removes a CICSplex SM or CICS definition from the data repository.

Notes:

1. Before you can update or remove a definition you must use the FETCH command to retrieve the appropriate resource table record from a result set.
2. For CICSplex SM definitions that have a CICSplex as their context (such as workload management or real-time analysis definitions), any changes you make are automatically distributed to all the CMASs involved in managing the CICSplex.

With each of these commands, you use the FROM option to supply a CPSM Definition or CICS Definition resource table record for the definition you are working with. The record must include all of the attributes identified in the resource table for the definition. If you do not want to specify certain optional attributes, you must set those fields to the appropriate null values for the attribute's data type (that is, zeroes or blank spaces).

As an alternative, when you are updating CICS definitions, you can use the RESULT and MODIFY options of the UPDATE command. These options enable you to modify multiple definitions at one time (this is the equivalent of issuing the ALTER action command from the CICSplex SM end-user interface).

To update CICS definitions, identify a result set that contains CICS Definition resource table records in the RESULT option. Then use the MODIFY option to specify the changes to be made to the definitions. MODIFY accepts a modification expression, as described in "Modifying resource attributes" on page 36.

The CHANGETIME and CREATETIME attributes

When you work with existing CICSplex SM or CICS definitions, keep in mind that the first 8 bytes of each record contain an attribute called CHANGETIME, which reflects the date and time at which the record was last modified. CICS Definition records also include a CREATETIME attribute, which is the date and time at which the definition was created.

The CHANGETIME and CREATETIME attributes are maintained internally by CICSplex SM; you should not attempt to modify these attribute values. When you update or remove a definition resource table record, the CHANGETIME and CREATETIME values you return to CICSplex SM must be the same values you received.

Using the PARM option

For most CICSplex SM and CICS definitions, all of the information needed to process an API request is included in the attributes of the resource table. Some definitions, however, allow you to supply optional data and some actually require additional data. For those definitions, you have to specify the PARM option on the appropriate API command:

- CREATE
- UPDATE
- REMOVE
- GET

The PARM option accepts a parameter expression, which is a character string that defines the parameters required for a definition to be processed.

For example, suppose you want to create an LNKSMSCG definition, which is a CICSplex SM definition that describes the association between a CICS system group and a monitor specification (MONSPEC). Before CICSplex SM can process your request, it must know how to handle other links that may be affected by the change. So when you issue the CREATE command, you must specify a parameter expression like this on the PARM option:

```
PARM('FORCE.')
```

which tells CICSplex SM that all CICS systems in the CICS system group are to inherit the new specification.

The PARM option is especially useful when working with CICS definitions. For each CICS Definition resource table there is another resource table that describes the definition's association with a resource group (RESGROUP), if one exists. For example, the CONNDEF resource table represents a connection definition and the CONINGRP resource table represents an association between a connection definition and a resource group. The RESGROUP parameter provided with the CREATE and GET commands for CICS Definitions simplifies the processing of these records.

When you create a CICS Definition record, you can identify an existing resource group to which the definition should be added. To do this, use the PARM option to identify the resource group like this:

```
PARM('RESGROUP(resgroup).')
```

Using the RESGROUP parameter automatically creates an xxxINGRP record (such as a CONINGRP record), which describes the association between the CICS definition and its resource group.

When you use the GET command to request CICS Definition records from the data repository, you can select definitions according to the resource group to which they belong. To do this, use the PARM option to identify the resource group like this:

```
PARM('RESGROUP(resgroup).')
```

which tells CICSplex SM to select CICS definitions only from the specified resource group. If you do not use the PARM option, CICSplex SM selects definitions from all resource groups, according to the other criteria you specify on the GET command.

Note: For a complete list of the CREATE, UPDATE, REMOVE, and GET parameters required (or supported) by a given resource table, see the *CICSplex SM Resource Tables Reference*.

Asynchronous processing

Most CICSplex SM API commands normally function in a synchronous manner, where your program issues a request and then waits until command processing is complete. However, some of the API commands also support asynchronous processing. This allows you to request data or perform actions without waiting for the request to complete. This support also enables you to receive notification when events of interest occur in the CICSplex.

The API commands you can use to request asynchronous processing are:

asynchronous processing

- LISTEN
- CANCEL
- GET
- PERFORM OBJECT
- PERFORM SET
- REFRESH
- SET.

The LISTEN command is, by its very nature, an asynchronous request because you are asking to be notified whenever a certain event occurs. The CANCEL command simply cancels an outstanding LISTEN request. The other commands can be used in either a synchronous or asynchronous manner. If you specify the NOWAIT option on any of these commands, the request is processed asynchronously.

The API commands you can use to monitor and receive the results of asynchronous processing are:

- ADDRESS
- RECEIVE.

Figure 12 illustrates the relationship of these commands to the API environment.

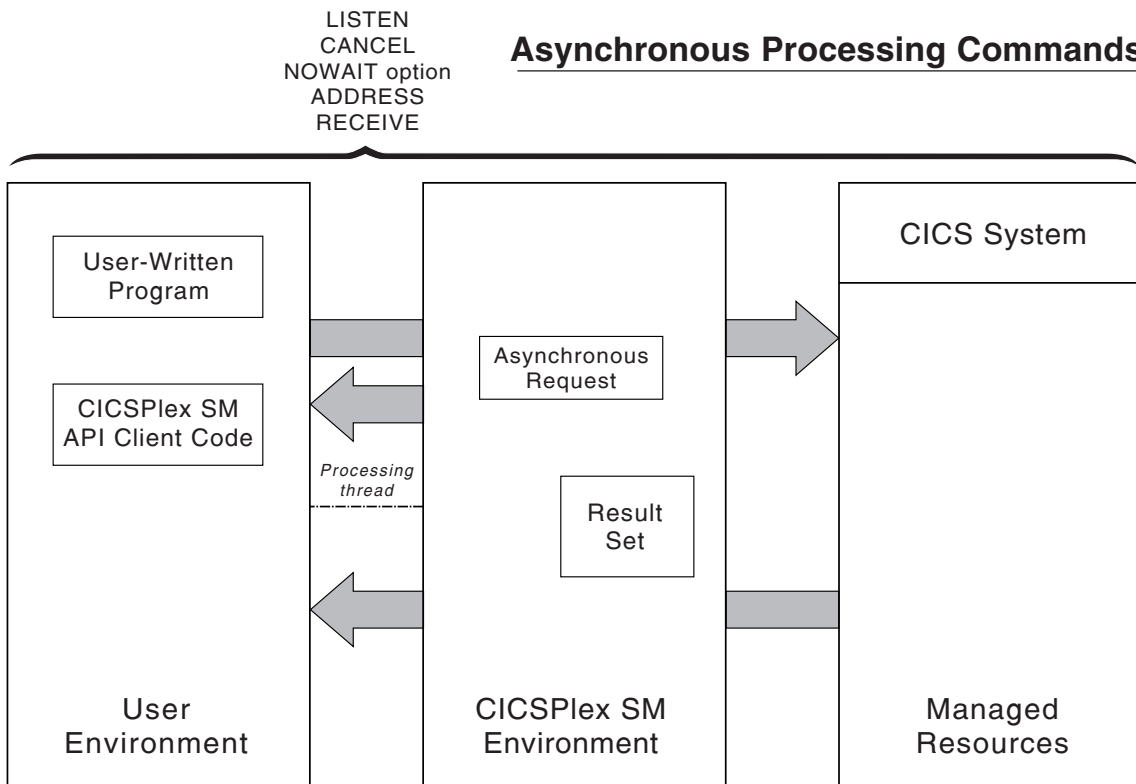


Figure 12. API commands for asynchronous processing

Using the LISTEN command

Many of the resources that are managed by CICSplex SM have the ability to notify the system when events occur that are considered significant to the CICSplex. Such events are not scheduled and cannot be anticipated, so a program designed to process these notifications must do so asynchronously. You can identify the event notifications you are interested in by using the LISTEN command.

The events that can be listened for are represented by resource tables with a type of CPSM Notification. For example, an EMASSICK notification is produced by a MAS when a condition occurs that adversely affects the health of the CICS system. For a list of the CPSM Notification resource tables, and complete descriptions of other resource table's, see *CICSplex SM Resource Tables Reference*.

When you issue a LISTEN command, the resulting notifications are added to an outstanding data queue for the API processing thread. The number of completed asynchronous requests, including event notifications and requests issued with the NOWAIT option, is reported by the SENTINEL option of the ADDRESS command. You can retrieve the event notifications by issuing a RECEIVE command.

Using the NOWAIT option

If you specify the NOWAIT option on a GET, PERFORM OBJECT, PERFORM SET, REFRESH, or SET command, the request does not complete processing immediately. Instead, the request is scheduled for processing, the command returns a RESPONSE value of SCHEDULED, and control returns to your program.

While the asynchronous request is executing, your program can perform other processing, even issuing another CICSplex SM API command. However, as long as a command is active, the result set it has been given to process is unavailable. A RESPONSE value of INUSE is returned if you try to access a result set that is still being processed by an asynchronous request.

An ASYNCREQ resource table record is produced when the asynchronous request completes. The number of completed asynchronous requests, including ASYNCREQ records that represent requests issued with the NOWAIT option, is reported by the SENTINEL option of the ADDRESS command. You can retrieve ASYNCREQ records by issuing a RECEIVE command.

The ASYNCREQ resource table includes much of the information that is normally returned by the command itself. Because control returns to your program before the command completes processing, that information is not available to the command. The information returned in the ASYNCREQ resource table includes:

- The command that was issued.
- The associated result set token.
- The RESPONSE and REASON values returned by the command.
- The diagnostic data normally returned in a FEEDBACK resource table record, if the RESPONSE value is not OK.
- A user-defined token that identifies the asynchronous request, if one was specified.

asynchronous processing

Note: To access the ASYNCREQ data from a REXX program, you can use either the CICSplex SM TPARSE command with the ASIS option or the REXX SUBSTR function.

Using tokens to identify a request

To keep track of the asynchronous requests your program issues, you can assign each request a unique identifying token. This allows your program to correlate LISTEN requests and requests made with the NOWAIT option with the results of a subsequent RECEIVE command. The CICSplex SM API makes no use of any tokens you define. User token values are simply held until the associated requests are complete and then returned to your program by the RECEIVE command. You can use any 1- to 4-character value as an identifying token. For example, you might specify:

- A literal constant
- An offset of a service routine
- The address of a data structure.

Using the ADDRESS command

When you issue a CONNECT command and an API processing thread is established, two control fields are created in the MVS/ESA address space or CICS system where the program is running. By requesting the addresses of these thread control fields, you can determine if asynchronous output is available without the need for polling or suspending processing.

You can use the ADDRESS ECB() SENTINEL() command to request the addresses of these fields:

ECB

The ECB is posted by the API each time an asynchronous request completes and is added to the thread's outstanding data queue. With the ECB address, you can:

- Test the appropriate MVS/ESA post bits to determine if output is available.
- Issue an MVS/ESA WAIT command in a batch, TSO, or NetView program.
- Issue an EXEC CICS WAITCICS or WAIT EXTERNAL command in a CICS program.

The ECB field is cleared whenever the counter value in the SENTINEL field reaches 0.

SENTINEL

The sentinel is a 4-byte counter of completed asynchronous requests associated with the thread.

The sentinel value increases each time an asynchronous request completes.

Examples of completed asynchronous requests include:

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

The sentinel value decreases when a RECEIVE command is issued.

Notes:

1. You should use the ADDRESS command before issuing the RECEIVE command. If the sentinel value is 0, it means there are no completed asynchronous requests to be received.
2. Because of the nature of asynchronous processing, the sentinel value may understate the actual number of outstanding requests at any point in time. When processing multiple asynchronous requests, you should issue the RECEIVE IMMEDIATE command until a response of NODATA is returned to ensure that all output has been received.

Using the RECEIVE command

You can use the RECEIVE command to determine if any of the asynchronous requests you issued have completed. RECEIVE returns the output from those requests. 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.

Note: Before you issue the RECEIVE command, you should issue the ADDRESS command and check the SENTINEL value to determine if there are any outstanding asynchronous requests to be received. If the sentinel value is 0, there are no outstanding asynchronous requests to be received.

As an example, your program might issue a LISTEN command and a GET command with the NOWAIT operand on the same API thread. Then, in response to a RECEIVE command, you would receive either an ASYNCREQ resource table record for the GET command or a resource table record associated with the event you were listening for.

Alternatively, you can use multiple API threads to separate the output returned by subsequent RECEIVE commands. For example, you might create one thread and use it only for receiving event notifications from the LISTEN command. You might also create another thread for use by other API functions. In this way, you can control what output is returned by the RECEIVE commands issued against each thread.

Another reason you might want to create multiple API threads is because each thread can have only 256 asynchronous requests outstanding at one time. If your program issues a large number of asynchronous requests on a single API thread, you should issue the RECEIVE command at regular intervals. If a processing thread reaches its maximum of 256, asynchronous requests are discarded and are not processed.

By default, the RECEIVE command waits until asynchronous output is available before returning control to your program. This means processing is suspended until an asynchronous request completes. As an alternative to waiting indefinitely, you can specify one of these options on the RECEIVE command:

DELAY(data-value)

Checks for asynchronous output, waits the specified number of seconds for output to become available, and then returns control to the processing thread, with or without output.

IMMEDIATE

Checks for asynchronous output and then immediately returns control to the processing thread, whether or not any output is available.

Using CICSplex SM tokens

Many of the CICSplex SM API commands are interrelated; you use them in conjunction with each other to accomplish the objectives of your program. For example, you issue a GET command to build a result set and then issue a FETCH command to access the resource table records in that result set.

To correlate the results of various operations with subsequent requests that you make, CICSplex SM assigns 4-byte tokens to the following objects of the API environment:

- Processing threads
- Result sets
- Filters
- Views
- LISTEN requests.

So, for example, each processing thread has a unique, 4-byte identifying token. You must specify a thread token on each API command that your program issues to identify the thread where it should be processed. Likewise, once a result set or filter is created, you refer to it on subsequent commands by supplying the token value assigned to it by CICSplex SM. And each LISTEN request is given a token so that you can cancel the request using the CANCEL command.

Notes:

1. CICSplex SM assigns a token to views for internal use only. Externally, you refer to a view by the name which you assigned to it.
2. There is a limit to the number of CICSplex SM tokens available to each processing thread. In general, the number of result sets, filters, views, and LISTEN requests created on a processing thread cannot exceed 255.

Token values are not only unique for individual objects, but the structure of the tokens varies by object type. So a thread token cannot be mistaken by CICSplex SM for any other type of token. If you specify an invalid token (such as, a result set token on the FILTER option), you receive a RESPONSE value of INVALIDPARM.

Using meta-data resource tables

The GETDEF command is used to obtain records describing the structure of the CICSplex SM managed objects, including general characteristics, valid actions, and object attributes.

The OBJECT option of the GETDEF command identifies the type of meta-data to be retrieved. The contents of the following meta-data resource tables are described below:

- ATTR
- ATTRAVA
- METADESC

- OBJECT
- OBJECT

ATTR

The ATTR resource table provides detailed information for a specific attribute of a managed object.

Attribute	Description
OBJECT	The name of the managed object to which the specific attribute belongs.
TABLEVER	The version of the table identified by the OBJECT attribute.
NAME	The name of the specific attribute. 1 to 12 characters in length.
LENGTH	The length of the data associated with the attribute. Not to be confused with the length of the ATTRattribute NAME.
OFFSET	The offset in the resource table at which the attribute data begins.
DATATYPE	The data type of the attribute data:
COMPID	CMPID
BINARY	Binary
RATE	Rate
PERCENT	Percent
SUM	Sum
RATIO	Ratio
AVERAGE	Average
TIMESTP	Time stamp
BIT	Bit string
TEXT	Text
CHAR	Character
EYUDA	EYUDA
CVDAS	Standard CVDA
CVDAT	Terminal CVDA
RESTYPE	Restype
DECIMAL	Packed Decimal
DATE	Date
ILABEL	Internal Label
HHMM	Binary Hours/Minutes
SCLOCK	CICS CMF SCHLOCK
INTUSEC	Interval in usec
INTMSEC	Interval in millisec
INT16US	Interval in 16usec

	INTSEC	Interval in Seconds
	INTTSTP	Interval Timestamp Delta
	DTGROUP	Date Time Group
	DECTSTP	Decimal Timestamp
	ADDRESS	Address
	CNUMERIC	Coded Numeric
	HIDCHAR	Non Display Character
	HEX	Hexadecimal
	SUMOPT	The default summary option used for the attribute:
	AVG	Average
	DIFF	Difference
	MIN	Minimum
	MAX	Maximum
	SUM	Summary
	LIKE	Like
	SETVALID	Whether or not the attribute may be set/modified: Y or N
	REQUIRED	Whether or not the attribute is required for CREATE: Y or N
	AVAAVAIL	Whether or not attribute value assertion information is available for the attribute: Y or N. When available, refer to the ATTR attributes:
		AVACOUNT
		Use the ATTRAVA resource table to obtain attribute value assertion information.
	CICSVALAVAIL	Whether or not CICS validity data is available: Y or N. When available, refer to the ATTR attributes:
		VALCICSESA
		VALCICSVSE
		VALCICSOS2
		VALCICSWNT
	HDRTXTAVAIL	Whether or not attribute header text is available: Y or N. When available, refer to the ATTR attributes:
		HDRTEXT
	VALSETAVAIL	Whether or not value set information is available: Y or N. When available, refer to the ATTR attributes:
		VALCOUNT
		Use the ATTRAVA resource table to obtain value set information.
	SOURCE	The source of the attribute data:
	V	Created by CPSM
	I	Acquired from CICS INQ

	S	Acquired from CICS STATS
	P	Acquired from CICS CMF data
KEY		Whether or not the attribute participates in the key of the managed object: 0 or n, where 0 means the attribute is not part of the key, and n means the part number of the key.
AVACOUNT		The number of attribute value assertions for the attribute. This value corresponds to the number of ATTRAVA resource table records available with a LISTTYPE value of AVA for the attribute. Only present if the AVAAVAIL attribute is Y.
VALCOUNT		The number of value set values for the attribute. This value corresponds to the number of ATTRAVA resource table records available with a LISTTYPE value of VALUE for the attribute. Only present if the VALSETAVAIL attribute is Y.
VALCICSESA		Flags indicating whether or not the attribute is valid in different versions of CICS/ESA: 1... X'80' CICS/MVS 2.1.2 .1.. X'40' CICS/ESA 3.3.0 ..1. X'20' CICS/ESA 4.1.0 ...1 X'10' CICS Transaction Server for OS/390 Release 1 1... X'08' CICS Transaction Server for OS/390 Release 21.. X'04' CICS Transaction Server for OS/390 Release 311 Reserved The attribute is not valid in the version of CICS if the bit is set on.
VALCICSVSE		Flags indicating whether or not the attribute is valid in different versions of CICS/VSE: 1... X'80' CICS/VSE 2.2.0 .1.. X'40' CICS/VSE 2.3.0 ..1. X'20' CICS/VSE 4.1.0 ...1 1111 Reserved The attribute is not valid in the version of CICS if the bit is set on.
VALCICSOS2		Flags indicating whether or not the attribute is valid in different versions of CICS OS/2: 1... X'80' CICS OS/2 2.0.1 .1.. X'40' CICS OS/2 3.0.0 ..1. X'20' CICS OS/2 3.1.0 ...1 1111 Reserved The attribute is not valid in the version of CICS if the bit is set on.
VALCICSWNT		Flags indicating whether or not the attribute is valid in different versions of CICS/NT: 1... X'80' CICS NT 5.0.0 .111 1111 Reserved

The attribute is not valid in the version of CICS if the bit is set on.

SETCICSESA

Flags indicating whether or not the attribute is modifiable in different versions of CICS/ESA:

1... ..	X'80'	CICS/MVS 2.1.2
.1.. ..	X'40'	CICS/ESA 3.1.0
..1. ..	X'20'	CICS/ESA 4.1.0
...1 ..	X'10'	CICS Transaction Server for OS/390 Release 1
.... 1..	X'08'	CICS Transaction Server for OS/390 Release 2
.... .1..	X'04'	CICS Transaction Server for OS/390 Release 3
.... ..11		Reserved

The attribute is not modifiable in the version of CICS if the bit is set on.

SETCICSVSE

Flags indicating whether or not the attribute is modifiable in different versions of CICS/VSE:

1... ..	X'80'	CICS/VSE 2.2.0
.1.. ..	X'40'	CICS/VSE 2.3.0
..1.	X'20'	CICS/VSE 4.1.0
...1 1111		Reserved

The attribute is not modifiable in the version of CICS if the bit is set on.

SETCICSOS2

Flags indicating whether or not the attribute is modifiable in different versions of CICS OS/2:

1... ..	X'80'	CICS OS/2 2.0.1
.1.. ..	X'40'	CICS OS/2 3.0.0
..1.	X'20'	CICS OS/2 3.1.0
...1 1111		Reserved

The attribute is not modifiable in the version of CICS if the bit is set on.

SETCICSWNT

Flags indicating whether or not the attribute is modifiable in different versions of CICS/NT:

1... ..	X'80'	CICS NT 5.0.0
.111 1111		Reserved

The attribute is not modifiable in the version of CICS if the bit is set on.

IGNVALUE

The value that signifies Not Applicable or Ignore for the attribute.

LOWVALUE

The lowest value allowed in the range of valid values for the attribute.

HIGHVALUE

The highest value allowed in the range of valid values for the attribute.

HDRTEXT

The header text of the attribute. Only present if the HDRTXTAVAIL attribute value is Y.

DESC	The description of the attribute.
DEFAULT	The default value for the attribute, if any.
UCHAR	Whether or not the attribute value is uppercase: Y or N.

ATTRAVA

The ATTRAVA resource table provides attribute value information for a specific attribute of a managed object.

Information in this resource table is only available when the AVAAVAIL or VALSETAVAIL attributes of the ATTR resource table have a value of Y.

Attribute	Description						
OBJECT	The name of the managed object to which the specific attribute belongs.						
TABLEVER	The version of the table identified by the OBJECT attribute.						
NAME	The name of the specific attribute. 1 to 12 characters in length.						
AVAValue	A value for the attribute.						
LISTTYPE	Indicates if the AVAValue data is an attribute value assertion or an attribute value list: <table> <tr> <td>AVA</td> <td>A value in an attribute value assertion list</td> </tr> <tr> <td>VALUE</td> <td>A value in an attribute value list</td> </tr> </table>	AVA	A value in an attribute value assertion list	VALUE	A value in an attribute value list		
AVA	A value in an attribute value assertion list						
VALUE	A value in an attribute value list						
IOTYPE	Indicates whether the attribute value is used for input, output, or input and output operations: <table> <tr> <td>I</td> <td>Input</td> </tr> <tr> <td>O</td> <td>Output</td> </tr> <tr> <td>B</td> <td>Input and output</td> </tr> </table>	I	Input	O	Output	B	Input and output
I	Input						
O	Output						
B	Input and output						

METADESC

The METADESC resource table provides basic structure and layout information for a specific attribute of a managed object.

Attribute	Description										
NAME	The name of the specific attribute. 1 to 12 characters in length.										
LENGTH	The length of the data associated with the attribute. Not to be confused with the length of the METADESC attribute NAME.										
OFFSET	The offset in the resource table at which the attribute data begins.										
DATATYPE	The data type of the attribute data: <table> <tr> <td>0</td> <td>Component Identifier</td> </tr> <tr> <td>4</td> <td>Binary Numeric</td> </tr> <tr> <td>8</td> <td>Binary Derived Rate</td> </tr> <tr> <td>12</td> <td>Binary Derived Percent</td> </tr> <tr> <td>16</td> <td>Binary Derived Sum</td> </tr> </table>	0	Component Identifier	4	Binary Numeric	8	Binary Derived Rate	12	Binary Derived Percent	16	Binary Derived Sum
0	Component Identifier										
4	Binary Numeric										
8	Binary Derived Rate										
12	Binary Derived Percent										
16	Binary Derived Sum										

meta-data resource tables

	20	Binary Derived Ratio
	24	Binary Derived Average
	28	System/370 Timestamp
	32	Bit
	36	Text
	40	Character
	44	EYUDA
	48	CVDA Standard
	52	CVDA Terminal
	56	Resource Type
	60	Packed Decimal
	64	Date
	68	Internal Label Field
	72	Binary HHMM
	76	Interval Store Clock
	80	Interval Microseconds
	84	Interval Milliseconds
	88	Interval 16 Microseconds
	92	Interval Seconds
	96	Interval Store Clock delta
	100	Date Time Group
	104	Packed Decimal Timestamp
	108	Address
	112	Coded Numeric
	116	Non Display Character
	120	Hexadecimal
	INHERIT	Whether or not the attribute value is inheritable: Y or N. Valid only for CPSM Definition resource tables that participate in CICSplex inheritance.

OBJECT

The OBJECT resource table provides action information for a specific managed object.

Attribute	Description
OBJECT	The name of the managed object to which the specific action applies.
TABLEVER	The version of the table identified by the OBJECT attribute.
ACTION	The name of the action. 1 to 12 characters in length.

VALCICSESA	<p>Flags indicating whether or not the action is valid in different versions of CICS/ESA:</p> <table border="0"> <tr><td>1... ..</td><td>X'80'</td><td>CICS/MVS 2.1.2</td></tr> <tr><td>.1.. ..</td><td>X'40'</td><td>CICS/ESA 3.3.0</td></tr> <tr><td>..1.</td><td>X'20'</td><td>CICS/ESA 4.1.0</td></tr> <tr><td>...1</td><td>X'10'</td><td>CICS Transaction Server for OS/390 Release 1</td></tr> <tr><td>.... 1...</td><td>X'08'</td><td>CICS Transaction server for OS/390 Release 2</td></tr> <tr><td>.... .1..</td><td>X'04'</td><td>CICS Transaction Server for OS/390 Release 3</td></tr> <tr><td>.... ..11</td><td></td><td>Reserved</td></tr> </table> <p>The action is not valid in the version of CICS if the bit is set on.</p>	1... ..	X'80'	CICS/MVS 2.1.2	.1.. ..	X'40'	CICS/ESA 3.3.0	..1.	X'20'	CICS/ESA 4.1.0	...1	X'10'	CICS Transaction Server for OS/390 Release 1 1...	X'08'	CICS Transaction server for OS/390 Release 21..	X'04'	CICS Transaction Server for OS/390 Release 311		Reserved
1... ..	X'80'	CICS/MVS 2.1.2																				
.1.. ..	X'40'	CICS/ESA 3.3.0																				
..1.	X'20'	CICS/ESA 4.1.0																				
...1	X'10'	CICS Transaction Server for OS/390 Release 1																				
.... 1...	X'08'	CICS Transaction server for OS/390 Release 2																				
.... .1..	X'04'	CICS Transaction Server for OS/390 Release 3																				
.... ..11		Reserved																				
VALCICSVSE	<p>Flags indicating whether or not the action is valid in different versions of CICS/VSE:</p> <table border="0"> <tr><td>1... ..</td><td>X'80'</td><td>CICS/VSE 2.2.0</td></tr> <tr><td>.1.. ..</td><td>X'40'</td><td>CICS/VSE 2.3.0</td></tr> <tr><td>..1.</td><td>X'20'</td><td>CICS/VSE 4.1.0</td></tr> <tr><td>...1 1111</td><td></td><td>Reserved</td></tr> </table> <p>The action is not valid in the version of CICS if the bit is set on.</p>	1... ..	X'80'	CICS/VSE 2.2.0	.1.. ..	X'40'	CICS/VSE 2.3.0	..1.	X'20'	CICS/VSE 4.1.0	...1 1111		Reserved									
1... ..	X'80'	CICS/VSE 2.2.0																				
.1.. ..	X'40'	CICS/VSE 2.3.0																				
..1.	X'20'	CICS/VSE 4.1.0																				
...1 1111		Reserved																				
VALCICSOS2	<p>Flags indicating whether or not the action is valid in different versions of CICS OS/2:</p> <table border="0"> <tr><td>1... ..</td><td>X'80'</td><td>CICS OS/2 2.0.1</td></tr> <tr><td>.1.. ..</td><td>X'40'</td><td>CICS OS/2 3.0.0</td></tr> <tr><td>..1.</td><td>X'20'</td><td>CICS OS/2 3.1.0</td></tr> <tr><td>...1 1111</td><td></td><td>Reserved</td></tr> </table> <p>The action is not valid in the version of CICS if the bit is set on.</p>	1... ..	X'80'	CICS OS/2 2.0.1	.1.. ..	X'40'	CICS OS/2 3.0.0	..1.	X'20'	CICS OS/2 3.1.0	...1 1111		Reserved									
1... ..	X'80'	CICS OS/2 2.0.1																				
.1.. ..	X'40'	CICS OS/2 3.0.0																				
..1.	X'20'	CICS OS/2 3.1.0																				
...1 1111		Reserved																				
VALCICSWNT	<p>Flags indicating whether or not the action is valid in different versions of CICS/NT:</p> <table border="0"> <tr><td>1... ..</td><td>X'80'</td><td>CICS NT 5.0.0</td></tr> <tr><td>..11 1111</td><td></td><td>Reserved</td></tr> </table> <p>The action is not valid in the version of CICS if the bit is set on.</p>	1... ..	X'80'	CICS NT 5.0.0	..11 1111		Reserved															
1... ..	X'80'	CICS NT 5.0.0																				
..11 1111		Reserved																				

OBJECT

The OBJECT resource table provides detailed information for a specific managed object.

Attribute	Description
NAME	The name of the managed object. 1 to 8 characters in length.
NUMTBLVER	The number of different versions of the managed object which are known to exist.
HIGHTBLVER	The number of the highest version of the managed object.
RELTBLVER	The version of the managed object at the current CPSM release.
OWNERNAME	The name of the component which owns the managed object.
CREATREL	CPSM release at which the managed object was introduced.
QUERYREL	CPSM release of the querying CMAS.

meta-data resource tables

OBJTYPE	The object type of the managed object:
C	CICS Resource
M	Monitored CICS Resource
D	CPSM Definition
V	CPSM Resource
O	CPSM Metadata
N	CPSM Notification
R	CICS Resource Definition
L	CPSM Configuration Definition
CURTLVER	Version of the managed object at the current CONNECT version
CURNUMATTR	Number of attributes in the managed object at the current CONNECT version
CURSTGSI	External length of the managed object at the current CONNECT version
CURCPMREL	CPSM release when the version of the managed object at the current CONNECT version was created
CURVALRTA	Whether or not the managed object is valid for use with RTA: Y or N.
CURVALUTL	Whether or not the managed object is valid for use with the batch utility: Y or N.
CURGETVAL	Whether or not the managed object is valid for GET requests: Y or N.
CURSETVAL	Whether or not the managed object is valid for SET requests: Y or N.
CURCREVAL	Whether or not the managed object is valid for CREATE requests: Y or N.
CURUPDVAL	Whether or not the managed object is valid for UPDATE requests: Y or N.
CURREMVAL	Whether or not the managed object is valid for REMOVE requests: Y or N.
CURACTVAL	Whether or not the managed object has actions defined: Y or N. Use the OBJACT resource table to obtain action information.
CURVALESA	Flags indicating whether or not the managed object is valid in different versions of CICS/ESA:

1... ..	X'80'	CICS/MVS 2.1.2
.1.. ..	X'40'	CICS/ESA 3.3.0
..1.	X'20'	CICS/ESA 4.1.0
...1	X'10'	CICS Transaction Server for OS/390 Release 1
.... 1...	X'08'	CICS Transaction Server for OS/390 Release 2
.... .1..	X'04'	CICS Transaction Server for OS/390 Release 3
.... ..11		Reserved

The object is not valid in the version of CICS if the bit is set on.

CURVALVSE

Flags indicating whether or not the managed object is valid in different versions of CICS/VSE:

1... ..	X'80'	CICS/VSE 2.2.0
.1.. ..	X'40'	CICS/VSE 2.3.0
..1.	X'20'	CICS/VSE 4.1.0
...1 1111		Reserved

The object is not valid in the version of CICS if the bit is set on.

CURVALOS2

Flags indicating whether or not the managed object is valid in different versions of CICS OS/2:

1... ..	X'80'	CICS OS/2 2.0.1
.1.. ..	X'40'	CICS OS/2 3.0.0
..1.	X'20'	CICS OS/2 3.1.0
...1 1111		Reserved

The object is not valid in the version of CICS if the bit is set on.

CURVALWNT

Flags indicating whether or not the managed object is valid in different versions of CICS/NT:

1... ..	X'80'	CICS NT 5.0.0
.111 1111		Reserved

The object is not valid in the version of CICS if the bit is set on.

DESC

The description of the managed object.

Querying the CICSplex SM API exit

In a CICS LMAS environment the CICSplex SM API function is implemented via a task related user exit. CICS application programs can use the EXEC CICS INQUIRE EXITPROGRAM command to retrieve information about the CICSplex SM API task related user exit:

```
EXEC CICS INQUIRE EXITPROGRAM(EYU9XLAP)
        CONNECTST(cvda)
        QUALIFIER(data-area)
```

In CICS systems that support the CONNECTST and QUALIFIER keywords of the INQUIRE EXITPROGRAM command, CONNECTST returns a CVDA indicating the status of the CICSplex SM API task related user exit (see the *CICS System Programming Reference* for more details on INQUIRE EXITPROGRAM), and QUALIFIER returns the name of the CICSplex to which the LMAS is connected.

Chapter 3. Writing an EXEC CPSM program

This chapter describes how to use the CICSplex SM command-level interface to write an API program. It describes the language-specific copy books that are supplied for each CICSplex SM resource table. It also describes the translation process and the compile, link-edit, and run-time considerations for each environment.

Using the resource table copy books

The CICSplex SM API accepts and returns resource data in the form of records that contain the resource attributes. For example, if you issue a FETCH command against a result set containing LOCTRAN resource table records, the API returns all the attributes for a given transaction in a single record. Your program must identify an area of storage to receive the resource table records.

Note: This method of returning data differs from the EXEC CICS system programming interface, where you must fetch each attribute of a resource individually.

To simplify the use of these resource table records, CICSplex SM provides a set of copy books for each resource table that you can access from an API program. By including these copy books in your program, you can access the resource table data in the appropriate structure and format for the language you are using.

How to access the copy books

The copy books are installed as part of the CICSplex SM installation process. They are placed into the following CICS Transaction Server for OS/390 Release 3 libraries, according to the language they support:

Language	Library name
Assembler	CICSTS13.CPSM.SEYUMAC
PL/I	CICSTS13.CPSM.SEYUPL1
COBOL	CICSTS13.CPSM.SEYUCOB
C	CICSTS13.CPSM.SEYUC370

If you want to include the copy books in your program, make sure the appropriate library is available to the assemble or compile step.

Note: The CICSplex SM API uses variable names that begin with EYU. Make sure your program does not define variables or structures with variable names that are the same as variable names generated by the translator or declared in the resource table copy books. Also be careful that your program does not implicitly generate such variable names.

Copybook names and aliases

Each CICSplex SM resource table has a name that is unique within the product. In addition, a unique name is created for each copy book version of the resource table in each language. The copy book names take the form:

EYUtnnnn

where:

using the resource table copy books

t Identifies which language the copy book supports, as one of the following:

A	Assembler
P	PL/I
L	COBOL
C	C

nnnn Is a 4-character numeric resource table identifier.

For example:

EYUA0001 Is the Assembler DSECT for the CICSRRGN resource table.

EYUC2451 Is the C structured data type for the CMAS resource table.

To make the copy books easy to reference in your program, CICSplex SM provides alias support for the copy book names. The appropriate data set contains the following two entries for each resource table:

EYUtnnnn The resource table copy book name.

formname The format name alias, which is the resource table name as shown in *CICSplex SM Resource Tables Reference*.

So, using the previous example, the Assembler DSECT for the CICSRRGN resource table could be referred to as either EYUA0001 or its alias, CICSRRGN.

Copybook format

Each copy book contains a prologue that describes the resource table and its characteristics, including:

- Valid API operations
- Any parameters that are required for an operation
- Valid API actions
- CICS releases that do not support the resource table, if any.

A description is provided for each attribute of the resource table. In addition, the following information is provided for an attribute, if appropriate:

- Whether the attribute can be modified by a SET command
- CICS releases that do not support the attribute, if any
- CICS releases that do not allow the attribute to be modified, if any.

Copybook data characteristics

Each resource table that can be processed by an API program contains data values for each of its attributes. The attribute values are presented in an internal format that is appropriate for the data type and the environment in which the program is running:

- Standard System/390® data formats are used. No translation or formatting operations are performed on the attribute values.
- For programs written in C, variable-length character fields do not contain the zero-byte ending delimiter.
- The lengths of all resource table records are a multiple of 8 bytes. Each copy book contains a definition of the resource table length.
- System/390 boundary alignments are observed for all data types. That means all resource table records are maintained internally starting on

doubleword-aligned storage locations. Alignment fields are automatically generated in each copy book. These alignment fields, which contain binary zeros, have names like:

EYU_RSVnnnn

Make sure the data areas your program uses to send and receive resource table records have proper boundary alignment.

Supplied copy books

This section provides detailed information about the resource table copy books supplied for each language.

Assembler H copy books

Distributed in: CICSTS13.CPSM.SEYUMAC

Distributed as: DSECTs

Copybook names: EYUANnnnn

Note the following as you use the Assembler H copy books:

- DSECT and DS statements are used to describe the resource table.
- The DSECT name is the resource table format name (such as, EMASSTRT).
- The attribute names are a concatenation of the resource table format name and the attribute name, connected by an underscore (such as, EMASSTRT_CMASNAME).

Note: Because attribute names exceed 8 bytes in length, versions of Assembler other than Assembler H are not supported.

- EQU statements are used to describe the setting of indicator fields for bit, binary, and character values.
- The table length field is a concatenation of the resource table format name and the constant TBL_LEN, connected by an underscore (such as, EMASSTRT_TBL_LEN).

The resource table data types are defined using the data definition operands of the DS statement. The following data type definitions are used:

DS X - Bit, binary values greater than 8 bytes
 - Odd number binary values less than 8 bytes
 - Mixed character and binary data

DS H - 2-byte binary numeric values

DS F - 4-byte binary numeric values
 - 4-byte intervals

DS D - Time stamps and 8-byte intervals
 - 8-byte numeric values

DS P - Packed decimal data

DS C - Character data

Figure 13 is a representative extract of an Assembler H resource table copy book:

```

*-----*
* Name = EYUA2400 *
* Format Name = EMASSTR *
* Version = 0001 *
* Status = CPSMREL(0140) *
* Function = Base Table Structure generator *
* Format definition for this element = EMASSTR *
* Valid Operations = None *
* Valid Actions = None *
*-----*
EMASSTR          DSECT      Notify CICS System Start Event
EMASSTR_CMASNAME DS CL0008  CMAS Name
EMASSTR_PLEXNAME DS CL0008  CICSPlex Name
, VS COBOL
EMASSTR_CSYSNAME DS CL0008  CICS System Name
EMASSTR_MON_SPEC DS CL0008  Monitor Spec Name
EMASSTR_RTA_SPEC DS CL0008  Real Time Analysis Spec Name
EMASSTR_WLM_SPEC DS CL0008  Work Load Manager Spec Name
EMASSTR_STATUS   DS XL0001  Status
EMASSTR_STATUS_LOCAL EQU 128   Local MAS
EMASSTR_STATUS_REMOTE EQU 64    Remote MAS
EMASSTR_DYNROUTE DS XL0001  Dynamic Routing Mode
EMASSTR_DYNROUTE_ACTIVE EQU 1    Routing ACTIVE
EMASSTR_DYNROUTE_SUSPEND EQU 2    Routing SUSPENDED
EMASSTR_DYNTYPE   DS CL0003  Dynamic Routing Type
EMASSTR_DYNTYPE_WLMTOR EQU C'TOR' Routing TOR
EMASSTR_DYNTYPE_WLMAOR EQU C'AOR' Routing AOR
EMASSTR_DESC      DS CL0030  Description
EMASSTR_CSYSAPPL DS CL0008  CICS System VTAM APPLID
, VS COBOL
EMASSTR_EYU_RSV0015 DS XL0005  Alignment Padding
EMASSTR_MASSTART   DS D      MAS Start STCK Value
EMASSTR_TMEZONEO   DS XL0001  Time Zone Offset
EMASSTR_TMEZONE    DS CL0001  Time Zone
EMASSTR_EYU_RSV0019 DS XL0002  Alignment Padding
EMASSTR_DAYLIGHTSV DS F      DayLight savings in effect
EMASSTR_SYSID      DS CL0004  MAS System Id
EMASSTR_OPSYSREL   DS CL0004  MAS Op Sys Release
EMASSTR_MVSNAME    DS CL0004  MVS System Name
EMASSTR_JOBNAME    DS CL0008  MAS Job Name
EMASSTR_CECNAME    DS CL0008  CEC Name
EMASSTR_SYSPLEX    DS CL0008  SYSPlex Name
, VS COBOL
EMASSTR_EYU_RSV0257 DS XL0004  Alignment Padding
EMASSTR_TBL_LEN    EQU 152    Current Table size

```

Figure 13. Sample Assembler H copy book

PL/I copy books

Distributed in: CICSTS13.CPSM.SEYUPL1

Distributed as: Based structures

Copybook names: EYUPnnnn

Note the following as you use the PL/I copy books:

- The variable EYUPTPTR must be explicitly declared as follows:
DCL EYUPTPTR POINTER;

- The structure level 1 name is the resource table format name (such as, EMASSTRT).
- The attribute names are used as subordinate level names.
- For attributes that describe bit indicators, subordinate structure levels are used. Each bit indicator is assigned a unique name.
- All other indicator attributes result in constant declarations being generated at the end of the resource table. These constants can be used for assignment or evaluation of the attribute. The constant name is a concatenation of the resource table name, the attribute name, and the indicator name, connected by underscores (such as, EMASSTRT_DYNROUTE_ACTIVE).
- The table length field is a concatenation of the resource table format name and the constant TBL_LEN, connected by an underscore (such as, EMASSTRT_TBL_LEN).

The resource table data types are mapped into the valid set of PL/I data types. However, exact mapping is not always possible. The resource table data types are mapped as follows:

```

    BIT(8) ALIGNED - 1-byte binary numeric values

    FIXED BIN(15) - 2-byte binary numeric values

    FIXED BIN(31) - 4-byte binary numeric values
                  - 4-byte intervals

(2) FIXED BIN(31) - Time stamps and 8-byte intervals
                  - 8-byte binary numeric values
                    (an array of two fullwords)

    FIXED DEC(n) - Packed decimal data

    CHAR(nnnn) - Character data
                - Binary values greater than 8 bytes
                - Odd number binary values less than 8 bytes
    
```

Figure 14 is a representative extract of a PL/I resource table copy book:

```

/*-----*/
/* Name = EYUP2400 */
/* Format Name = EMASSTRT */
/* Version = 0001 */
/* Status = CPSMREL(0140) */
/* Function = Base Table Structure generator */
/* Format definition for this element = EMASSTRT */
/* Valid Operations = None */
/* Valid Actions = None */
/*-----*/

DCL 01 EMASSTRT BASED(EYUPTPTR), /* Notify CICS System Start Event*/
    02 CMASNAME CHAR(0008),
        /* CMAS Name */
    02 PLEXNAME CHAR(0008),
, VS COBOL
    /* CICSplex Name */
    02 CSYSNAME CHAR(0008),
        /* CICS System Name */
    02 MON_SPEC CHAR(0008),
        /* Monitor Spec Name */
    02 RTA_SPEC CHAR(0008),
        /* Real Time Analysis Spec Name */
    02 WLM_SPEC CHAR(0008),
        /* Work Load Manager Spec Name */

    02 STATUS,
        /* Status */
    03 LOCAL BIT(1) UNALIGNED,
        /* Local MAS */
    03 REMOTE BIT(1) UNALIGNED,
        /* Remote MAS */
    03 RSVD0003 BIT(1) UNALIGNED,
        /* Reserved */
    03 RSVD0004 BIT(1) UNALIGNED,
        /* Reserved */
    03 RSVD0005 BIT(1) UNALIGNED,
        /* Reserved */
    03 RSVD0006 BIT(1) UNALIGNED,
        /* Reserved */
    03 RSVD0007 BIT(1) UNALIGNED,
        /* Reserved */
    03 RSVD0008 BIT(1) UNALIGNED,
        /* Reserved */

```

Figure 14 (Part 1 of 3). Sample PL/I copy book

```

02 DYNROUTE      BIT(8) ALIGNED,
/* Dynamic Routing Mode                                */
02 DYNTYPE       CHAR(0003),
/* Dynamic Routing Type                                */
02 DESC          CHAR(0030),
/* Description                                          */
02 CSYSAPPL      CHAR(0008),
, VS COBOL
/* CICS System VTAM APPLID                             */
, VS COBOL
02 EYU_RSV0015   CHAR(0005),
/* Alignment Padding                                   */
02 MASSTART(2)   FIXED BIN(31),
/* MAS Start STCK Value                                */
02 TMEZONE0      BIT(8) ALIGNED,
/* Time Zone Offset                                    */
02 TMEZONE       CHAR(0001),
/* Time Zone                                           */
02 EYU_RSV0019   CHAR(0002),
/* Alignment Padding                                   */
02 DAYLGHTSV     FIXED BIN(31),
/* DayLight savings in effect                          */
02 SYSID         CHAR(0004),
/* MAS System Id                                       */
02 OPSYSREL      CHAR(0004),
/* MAS Op Sys Release                                  */
02 MVSNAME       CHAR(0004),
/* MVS System Name                                     */
02 JOBNAME       CHAR(0008),
/* MAS Job Name                                        */
02 CECNAME       CHAR(0008),
/* CEC Name                                            */
02 SYSPLEX       CHAR(0008),
, VS COBOL
/* SYSPlex Name                                        */
02 EYU_RSV0257   CHAR(0004);
/* Alignment Padding                                   */

```

Figure 14 (Part 2 of 3). Sample PL/I copy book

```
/*-----*/
/*                                          */
/* EMASSTRT Constants for Table          */
/*                                          */
/*-----*/
DCL EMASSTRT_DYNROUTE_ACTIVE BIT(8) ALIGNED STATIC INIT('01'BX);
    /* Routing ACTIVE                               */
DCL EMASSTRT_DYNROUTE_SUSPEND BIT(8) ALIGNED STATIC INIT('02'BX);
    /* Routing SUSPENDED                           */
DCL EMASSTRT_DYNTYPE_WLMTOR CHAR(3) STATIC INIT('TOR');
    /* Routing TOR                                 */
DCL EMASSTRT_DYNTYPE_WLMAOR CHAR(3) STATIC INIT('AOR');
    /* Routing AOR                                 */
DCL EMASSTRT_TBL_LEN FIXED BIN(15) STATIC INIT(152);
```

Figure 14 (Part 3 of 3). Sample PL/I copy book

VS COBOL II copy books

Distributed in: CICSTS13.CPSM.SEYUCOB

Distributed as: Structures

Copybook names: EYULnnnn

Note the following as you use the VS COBOL II copy books:

- The structure level 1 name is the resource table format name (such as, EMASSTRT).
- The attribute names are used as subordinate level names.
- For attributes that describe indicators, subordinate 88 levels are used. Each indicator is assigned a unique name. Hexadecimal literals are used to describe the content of the indicator setting.
- By default, CICSplex SM attribute names are formed with a connecting underscore character, as in WLM_SPEC. However, since VS COBOL II syntax does not support underscores, all attribute names that contain underscores are converted in the copy books to use hyphens, as in WLM-SPEC. When attribute names are passed to the API, they must contain the underscore character, not the hyphen used by VS COBOL II.
- All the resource tables use apostrophe characters as literal delimiters. When you translate or compile your program with a supplied copy book, you must specify the APOST option. Otherwise, you will receive VS COBOL II warning messages.
- VS COBOL II reserves many words for its own use. Some of the CICSplex SM resource table and attribute names conflict with these reserved words. To prevent such a conflict, any CICSplex SM name that conflicts with a VS COBOL II reserved word is modified by adding a suffix of -R. For example, the name of the CONNECT resource table becomes CONNECT-R and the name of the STATUS attribute becomes STATUS-R. The comment area for a name that would conflict with VS COBOL II shows the description "-- RESERVED WORD --". When resource table or attribute names are passed to the API, they must not include the -R suffix.
- VS COBOL II does not support duplicate names at different levels in the same data structure. Some of the CICSplex SM attribute names are the same as resource table names. To prevent a duplicate name problem, any attribute

name that is the same as a resource table name is modified by adding a suffix of -A. For example, the name of the DSNAME attribute becomes DSNAME-A. The name of the DSNAME resource table remains unchanged. The comment area for an attribute that has the same name as a resource table shows the description "-- RESERVED WORD --". When attribute names are passed to the API, they must not include the -A suffix.

- The table length field is a concatenation of the resource table format name and the constant TBL-LEN, connected by a hyphen (such as, EMASSTRT-TBL-LEN).

The resource table data types are mapped into the valid set of VS COBOL II data types. However, exact mapping is not always possible. The resource table data types are mapped as follows:

```

PIC S9(0004) USAGE BINARY - 2-byte binary numeric values

PIC S9(0008) USAGE BINARY - 4-byte binary numeric values
                        - 4-byte intervals

PIC S9(0016) USAGE BINARY - Time stamps and 8-byte intervals
                        - 8-byte binary numeric values

PIC S9(nnnn) USAGE PACKED-DECIMAL - Packed decimal data

PIC X(0001) - 1-byte binary and bit indicators

PIC X(nnnn) - Character data
            - Binary values greater than 8 bytes
            - Odd number binary values less than
              8 bytes
    
```

Figure 15 is a representative extract of a VS COBOL II resource table copy book:

using the resource table copy books

```
* -----*
* Name = EYUL2400 *
* Format Name = EMASSTR *
* Version = 0001 *
* Status = CPSMREL(0140) *
* Function = Base Table Structure generator *
* Format definition for this element = EMASSTR *
* Valid Operations = None *
* Valid Actions = None *
* -----*
01 EMASSTR.
* Notify CICS System Start Event
  02 CMASNAME PIC X(0008).
* CMAS Name
  02 PLEXNAME PIC X(0008).
, VS COBOL
* CICSplex Name
  02 CSYSNAME PIC X(0008).
* CICS System Name
  02 MON-SPEC PIC X(0008).
* Monitor Spec Name
  02 RTA-SPEC PIC X(0008).
* Real Time Analysis Spec Name
  02 WLM-SPEC PIC X(0008).
* Work Load Manager Spec Name
  02 STATUS-R PIC X(0001).
* Status -- RESERVED WORD --
  88 LOCAL VALUE X'80'.
* Local MAS
  88 REMOTE VALUE X'40'.
* Remote MAS
  02 DYNROUTE PIC X(0001).
* Dynamic Routing Mode
  88 ACTIVE VALUE X'01'.
* Routing ACTIVE
  88 SUSPEND VALUE X'02'.
* Routing SUSPENDED
  02 DYNTYPE PIC X(0003).
* Dynamic Routing Type
  88 WLMTOR VALUE 'TOR'.
* Routing TOR
  88 WLMAOR VALUE 'AOR'.
* Routing AOR
  02 DESC PIC X(0030).
* Description
  02 CSYSAPPL PIC X(0008).
, VS COBOL
* CICS System VTAM APPLID
, VS COBOL
  02 EYU-RSV0015 PIC X(0005).
```

Figure 15 (Part 1 of 2). Sample VS COBOL II copy book

```

* Alignment Padding
  02 MASSTART      PIC S9(0016) USAGE BINARY.
* MAS Start STCK Value
  02 TMEZONE0      PIC X(0001).
* Time Zone Offset
  02 TMEZONE       PIC X(0001).
* Time Zone
  02 EYU-RSV0019   PIC X(0002).
* Alignment Padding
  02 DAYLGHTSV     PIC S9(0008) USAGE BINARY.
* DayLight savings in effect
  02 SYSID         PIC X(0004).
* MAS System Id
  02 OPSYSREL      PIC X(0004).
* MAS Op Sys Release
  02 MVSNAME       PIC X(0004).
* MVS System Name
  02 JOBNAME       PIC X(0008).
* MAS Job Name
  02 CECNAME       PIC X(0008).
* CEC Name
  02 SYSPLEX       PIC X(0008).
, VS COBOL
* SYSPlex Name
  02 EYU-RSV0257   PIC X(0004).
* Alignment Padding
* -----*
*
* EMASSTRT Constants for Table
*
* -----*
01 EMASSTRT-TBL-LEN PIC S9(4) USAGE BINARY VALUE 152.

```

Figure 15 (Part 2 of 2). Sample VS COBOL II copy book

C/370 copy books

Distributed in: CICSTS13.CPSM.SEYUC370

Distributed as: Structured data types

Copybook names: EYUCnnnn

Note the following as you use the C/370 copy books:

- Typedef statements are used to describe the resource table.
- The structure name is the resource table format name (such as, EMASSTRT).
- The attribute names are used as subordinate names.
- For attributes that describe bit indicators, #define statements are generated at the end of the resource table. Each #define statement identifies a single indicator value. These constants can be used for assignment or evaluation of the attribute. The constant name is a concatenation of the resource table name, the attribute name, and the indicator name, connected by underscores (such as, EMASSTRT_DYNROUTE_ACTIVE).
- The copy books use trigraphs, which are multi-character combinations, to represent square brackets.
- Any variable-length data that you send to the API must be padded with blanks to the end of the field. The API does not insert the zero-byte ending delimiter.

using the resource table copy books

- The table length field is a concatenation of the resource table format name and the constant TBL_LEN, connected by an underscore (such as, EMASSTR_TBL_LEN).

The resource table data types are mapped into the valid set of C data types. However, exact mapping is not always possible. The resource table data types are mapped as follows:

```
char - 1-byte binary numeric values
short int - 2-byte binary numeric values
long - 4-byte binary numeric values
      - 4-byte intervals
long 2 - Time stamps and 8-byte intervals
        8- byte binary numeric values
        (an array of two fullwords)
char nnnn - Packed decimal data
char nnnn - Character data
           - Binary values greater than 8 bytes
           - Odd number binary values less than 8 bytes
```

Figure 16 on page 67 is a representative extract of a C/370 resource table copy book:

```

/*-----*/
* Name = EYUC2400                                     *
* Format Name = EMASSTRT                             *
* Version = 0001                                     *
* Status = CPSMREL(0140)                             *
* Function = Base Table Structure generator           *
* Format definition for this element = EMASSTRT       *
* Valid Operations = None                            *
* Valid Actions = None                               *
/*-----*/
typedef struct EMASSTRT {
char    CMASNAME??(8??);    /* CMAS Name          */
char    PLEXNAME??(8??);    /* CICSplex Name      */
, VS COBOL
char    CSYSNAME??(8??);    /* CICS System Name   */
char    MON_SPEC??(8??);    /* Monitor Spec Name  */
char    RTA_SPEC??(8??);    /* Real Time Analysis Spec Name */
char    WLM_SPEC??(8??);    /* Work Load Manager Spec Name */
char    STATUS;             /* Status             */
char    DYNROUTE;           /* Dynamic Routing Mode */
char    DYNTYPE??(3??);     /* Dynamic Routing Type */
char    DESC??(30??);       /* Description         */
char    CSYSAPPL??(8??);    /* CICS System VTAM APPLID */
, VS COBOL
char    EYU_RSV0015??(5??); /* Alignment Padding  */
long    MASSTART??(2??);    /* MAS Start STCK Value */
char    TMEZONEO;           /* Time Zone Offset   */
char    TMEZONE;            /* Time Zone           */
char    EYU_RSV0019??(2??); /* Alignment Padding  */
long    DAYLGHTSV;          /* DayLight savings in effect */
char    SYSID??(4??);        /* MAS System Id      */
char    OPSYSREL??(4??);     /* MAS Op Sys Release */
char    MVSNAME??(4??);      /* MVS System Name    */
char    JOBNAME??(8??);      /* MAS Job Name        */
char    CECNAME??(8??);     /* CEC Name            */
char    SYSPLEX??(8??);     /* SYSplex Name       */
, VS COBOL
char    EYU_RSV0257??(4??); /* Alignment Padding  */
} EMASSTRT;

```

Figure 16 (Part 1 of 2). Sample C/370 copy book

```
/*-----*
 *
 * EMASSTRT Defines for Table
 *
 *-----*/
#define EMASSTRT_STATUS_LOCAL      128
#define EMASSTRT_STATUS_REMOTE    64
#define EMASSTRT_DYNROUTE_ACTIVE  1
#define EMASSTRT_DYNROUTE_SUSPEND 2
#define EMASSTRT_DYNTYPE_WLMTOR   "TOR"
#define EMASSTRT_DYNTYPE_WLMAOR   "AOR"
#define EMASSTRT_TBL_LEN          152
```

Figure 16 (Part 2 of 2). Sample C/370 copy book

Language and environment considerations

This section describes various language and environment considerations that you should keep in mind when writing a CICSplex SM API program. Note that all of the usual language considerations that apply to the various environments (CICS, MVS/ESA batch, TSO, and NetView) also apply to CICSplex SM programs written to run in those environments.

Assembler H considerations

For Assembler H programs that run in an MVS/ESA batch, TSO, or NetView environment, you need to be aware of the following special considerations:

- Since the program does not execute in CICS, do not use the DFHEIENT or DFHEIRET macros. Instead, use the CICS translator options NOEPILOG, NOPROLOG, and NOSYSEIB.
- You must explicitly code the DFHEISTG and DFHEIEND macros to provide the required work areas for EXEC CPSM commands. Your program is responsible for acquiring storage for the DFHEISTG area and setting up any necessary base registers prior to making any EXEC CPSM calls. This storage can be acquired dynamically using local GETMAIN services or, if the program is nonreentrant, the storage can be defined directly in the program area. Reentrant programs are recommended if there is any possibility of the program being used concurrently in the same address space.
- You must make the appropriate CICS macro library available in the SYSLIB concatenation for the Assembler step. The DFHEISTG, DFHEIEND, and DFHSCALL macros are fetched from this library.

PL/I considerations

For PL/I programs, you need to be aware of the following special considerations:

- The variable EYUPTPTR must be explicitly declared as follows:
DCL EYUPTPTR POINTER;

NetView considerations

If you plan to run C/370 programs under NetView, you need to be aware of the following special considerations:

- Depending on which resource tables you access, you may encounter some name conflicts between the CICSplex SM #define statements for resource table attributes and the standard NetView #define statements. For example, the NetView statement #include "dsic.h" generates the following define statement:

```
#define COMMAND "COMMAND "
```

Some of the CICSplex SM resource tables use COMMAND as an attribute name. If you use #include "dsic.h" as supplied by NetView, the resource table attribute names are converted and cannot be processed by CICSplex SM.

One way of handling any potential conflicts is to undefine the COMMAND value, like this:

```
#include "dsic.h"
#undef COMMAND
#include "feedback.h"
.
.
.
```

If you want to, you can also redefine the COMMAND value using a new name that does not conflict with any resource table attribute name, like this:

```
#include "dsic.h"
#undef COMMAND
#define XCOMMAND "COMMAND "
#include "feedback.h"
.
.
.
```

User-replaceable programs

The CICSplex SM API cannot be used from within the user-replaceable programs EYU9XESV and EYU9WRAM.

CICS Global User exit programs

The CICSplex SM API may be used from within the CICS XICEREQ Global User Exit program. You must avoid recursion within the CICSplex SM API program and the exit should not delay any requests issued by CPSM related tasks.

The use of the CICSplex SM API from within other CICS Global User Exit points is not recommended as the results are unpredictable.

Status programs

The CICSplex SM API cannot be used from within a program that is invoked through the STATDEF view. Where access to the API is required, you must start another task and invoke the API from the new task.

Translating your program

For programs written using the command-level interface, you must use a language translator to interpret the source program for the API. Any external program that contains EXEC CPSM commands must be processed by the appropriate version of the CICS/ESA command level translator.

The following versions of the CICS translator support EXEC CPSM commands:

- CICS/ESA 4.1 with APAR PN73812
- CICS TS for OS/390

Notes:

1. If you are using the CICS/ESA 4.1 version of the translator, make sure the appropriate APAR has been applied before you attempt to translate your program.
2. If you are using Business Application Services (BAS) to create CICS resource definitions, be sure to use the appropriate version of the translator for the definitions you are creating. That is, if you want to create CICS TS for OS/390 resource definitions, you must use the translator that is distributed with that version of CICS.

Specifying the CPSM translator option

Because CICSplex SM uses the CICS/ESA translator, you can use your CICS translate JCL as a model for translating CICSplex SM API programs. You must specify one additional translator option, called CPSM, in order to translate CICSplex SM programs. The CPSM option can be specified by using either the PARM operand of the EXEC statement or a language-specific XOPTS options statement.

If your program also contains EXEC CICS commands, those commands are processed in the same translation step. The CICS translator inserts the necessary variable and invocation definitions required for proper execution of the program.

When using the CPSM API in a non-CICS environment, be sure to remove any CICS or SP translator options, and only specify the CPSM translator option.

As a result of the translation process, EXEC CPSM statements are replaced with language specific calls to an EXEC interface stub program.

Sample Assembler H translation

To specify the CPSM translator option, use either the PARM operand of the EXEC statement, like this:

```
//TRANSLAT EXEC PGM=DFHEAP1$,PARM='CPSM',REGION=4096K
```

or an XOPTS options statement, like this:

```
*ASM XOPTS(...CPSM)
```

Sample PL/I translation

To specify the CPSM translator option, use either the PARM operand of the EXEC statement, like this:

```
//TRANSLAT EXEC PGM=DFHEPP1$,PARM='CPSM',REGION=4096K
```

or an XOPTS options statement, like this:

```
*PROCESS XOPTS(...CPSM)
```

Sample VS COBOL II translation

To specify the CPSM translator option, use either the PARM operand of the EXEC statement, like this:

```
//TRANSLAT EXEC PGM=DFHECP1$,PARM='COBOL2,CPSM',REGION=4096K
```

or an XOPTS options statement, like this:

```
PROCESS XOPTS(...CPSM)
```

Note that when you translate a VS COBOL II program, you must specify both the CPSM and the COBOL2 translator options.

Sample C/370 translation

To specify the CPSM translator option, use either the PARM operand of the EXEC statement, like this:

```
//TRANSLAT EXEC PGM=DFHEDP1$,PARM='CPSM',REGION=4096K
```

or an XOPTS options statement, like this:

```
#pragma XOPTS(...CPSM)
```

Compiling your program

Compiling a CICSplex SM API program is similar to compiling a CICS program. You can use your CICS compile JCL as a model and then make the following modifications according to the language you are using.

Assembler H considerations

You can use only the following Assembler compiler to process CICSplex SM programs written in Assembler:

- Assembler H Compiler Version 2 (or later)

To assemble CICSplex SM programs, you must include a SYSLIB statement for the CICSTS13.CPSM.SEYUMAC macro library in your compile JCL, like this:

```
//ASM      EXEC PGM=IEV90,REGION=4096K
           .
           .
           .
//SYSLIB   DD DSN=CICSTS13.CPSM.SEYUMAC,DISP=SHR
           .
           .
           .
```

PL/I considerations

You can use only the following PL/I compiler to process CICSPlex SM programs written in PL/I:

- OS PL/I Optimizing Compiler Version 2.3 (or later)

To compile CICSPlex SM programs, you must include a SYSLIB statement for the CICSTS13.CPSM.SEYUPL1 macro library in your compile JCL, like this:

```
//COMPILE EXEC PGM=IEL0AA,REGION=1000K,  
//          PARM='OBJECT,MACRO,LIST'  
          .  
          .  
          .  
//SYSLIB   DD DSN=CICSTS13.CPSM.SEYUPL1,DISP=SHR  
          .  
          .  
          .
```

COBOL considerations

You can use only the following COBOL compiler to process CICSPlex SM programs written in COBOL:

- VS COBOL II Compiler Version 1.3.2 (or later)

To compile CICSPlex SM programs, you must include a SYSLIB statement for the CICSTS13.CPSM.SEYUCOB macro library in your compile JCL, like this:

```
//COMPILE EXEC PGM=IGYCRCTL,REGION=4096K  
          .  
          .  
          .  
//SYSLIB   DD DSN=CICSTS13.CPSM.SEYUCOB,DISP=SHR  
          .  
          .  
          .
```

C/370 considerations

You can use only the following compiler and library to process CICSPlex SM programs written in C/370:

- C/370 Compiler Version 2.1 (or later)
- C/370 Library Version 2.1 (or later).

To compile CICSPlex SM programs, you must include a SYSLIB statement for the CICSTS13.CPSM.SEYUC370 macro library in your compile JCL, like this:

```
//COMPILE EXEC PGM=EDCCOMP,REGION=4096K  
          .  
          .  
          .  
//SYSLIB   DD DSN=CICSTS13.CPSM.SEYUC370,DISP=SHR  
          .  
          .  
          .
```

Link editing your program

The CICS/ESA translator inserts a call to the CICSplex SM EXEC interface stub program. The stub entry name is not the name of an object or load module. Since CICSplex SM API programs can run in a variety of environments, the stub reference must be resolved to a module consistent with the intended usage. This resolution is performed at link-edit time using the INCLUDE linkage editor control statement.

You must link edit all program load modules with the correct CICSplex SM stub module for the environment where the program will run. To do this, specify one of the following stub modules in the INCLUDE statement:

EYU9AMSI For CICS/ESA programs. EYU9AMSI is supplied in the CICSSTS13.CPSM.SEYULOAD library.

EYU9ABSI For batch, TSO, or NetView programs. EYU9ABSI is supplied in the CICSSTS13.CPSM.SEYUAUTH library.

Each of these stub modules contains the appropriate entrypoint identifier. The services provided by the entrypoint are unique to the type of execution environment.

Note: You should not attempt to run a program identified as a CICS program in a batch environment. Likewise, batch programs are not suitable for running under CICS.

You can use your CICS link-edit JCL as a model for link editing CICSplex SM programs. Be sure to review the language-specific considerations in the remainder of this section and modify your JCL accordingly.

In addition, if your program contains EXEC CICS commands, you should review the link-edit considerations in the *Application Programming Guide* for your version of CICS. Likewise, if your program runs under NetView, you should refer to the NetView customization book for your programming language, either *Customization: Using Assembler*, or *Customization: Using PL/I and C*.

Assembler H considerations

Assembler load modules can reside in 24- or 31-bit storage and can be entered in either addressing mode.

To link edit an Assembler module to run with a CICSplex SM program, you must include a SYSLIB statement for the SEYULOAD load library in your link-edit step. This allows you to include the appropriate CICSplex SM stub module when link editing. For example:

run-time considerations

```
//LKED EXEC PGM IEWL,  
// PARM='XREF,LET,LIST,AMODE=ANY,RMODE=31',  
// REGION=4096K,COND=(7,LT,ASM)  
.  
.  
.  
//SYSLIB DD DSN=CICSTS13.CPSM.SEYULOAD,DISP=SHR  
.  
.  
.  
INCLUDE SYSLIB(userprog)  
INCLUDE SYSLIB(EYU9AMSI)  
NAME LMODNAME(R)
```

PL/I, VS COBOL II, and C/370 considerations

PL/I, VS COBOL II, and C/370 load modules can reside in 24- or 31-bit storage and can be entered in either addressing mode.

To link edit a module to run with a CICSplex SM program, you must include a SYSLIB statement for the SEYULOAD load library in your link-edit step. This allows you to include the appropriate CICSplex SM stub module when link editing. For example:

```
//LKED EXEC PGM=IEWL,  
// PARM='XREF,LET,LIST,AMODE=ANY,RMODE=31',  
// REGION=4096K,COND=(8,LE,COMPILE)  
.  
.  
.  
//SYSLIB DD DSN=CICSTS13.CPSM.SEYULOAD,DISP=SHR  
.  
.  
.  
INCLUDE SYSLIB(userprog)  
INCLUDE SYSLIB(EYU9AMSI)  
NAME LMODNAME(R)
```

Run-time considerations

- The run-time version of a CICSplex SM API program is equal to the level of the CMAS to which it connects:
 - For a program written to run as a CICS application, the run-time version is that of the CMAS to which the MAS is connected.
 - For a program written to run as a batch job or under NetView or TSO, the version is determined by the version of the CICSplex SM run-time module (EYU9AB00).

EYU9AB00 is distributed in CPSMnnn.SEYUAUTH, where nnn is the version of CICSplex SM. At run time, CICSplex SM must find EYU9AB00 in the STEPLIB, MVS linklist, or LPA library concatenation.
- The run-time version of a program must be greater than or equal to:
 - The version of the stub routine module (EYU9AxSI) with which the program was link edited.

- The value specified on the VERSION option of the CONNECT command.
- For programs written in PL/I, VS COBOL II, or C/370, a set of run-time libraries is shipped with the language compiler. To run a CICSplex SM program written in one of these languages, you must modify your environment startup procedure to reference the appropriate run-time libraries for the language.
- Before running any CICSplex SM program under CICS, make sure the program and its associated transaction are defined to CEDA. The program may be defined with an EXECKEY value of either User or CICS. The associated transaction may be defined with a TASKDATAKEY value of either User or CICS.

|
|
|
|
|

run-time considerations

Chapter 4. Dealing with exception conditions

This chapter describes the tools and techniques that are available for dealing with error conditions in a CICSplex SM API program.

Note: For information on additional diagnostic data that is available for an API program, refer to *CICSplex SM Problem Determination*.

Default CICSplex SM exception handling

The CICSplex SM API writes an exception trace, in the form of a user trace record, to the CICS trace data set. Resources available via the CICSplex SM API are not recoverable, and, therefore, resources updated prior to the exception are neither recovered nor are they available for backout by the application using EXEC CICS SYNCPOINT and EXEC CICS SYNCPOINT ROLLBACK commands.

Using the RESPONSE and REASON options

The RESPONSE and REASON options are required on each API command. You should specify these options as user-defined variables to receive the numeric response and reason values returned by a command. You can then convert the numeric values into more meaningful character equivalents. In general, RESPONSE describes the result of command processing and REASON further qualifies the response to certain commands.

Note: The TBUILD and TPARSE commands, which can be used only with the REXX run-time interface, do not use the RESPONSE and REASON options. The result of these REXX-specific processes is returned by their STATUS option. For more information, see Chapter 6, “REXX error handling” on page 97.

Types of responses

There are three types of responses that an API command can return:

- Normal
- Warning
- Error.

The character equivalents of the RESPONSE and REASON values that can be returned are given in the description of each command. For a summary of RESPONSE and REASON character values by command, see *CICSplex SM Application Programming Reference*. For a list of RESPONSE and REASON character values and their numeric equivalents, also see *CICSplex SM Application Programming Reference*.

Normal responses

A normal response indicates the API command completed processing successfully. The following values represent a normal response:

OK The command was successfully processed and control was returned to the program. There are no reasons associated with a response of OK.

SCHEDULED A command that was issued with the NOWAIT option has been scheduled for processing. The actual result of command processing is returned by the RECEIVE command in an ASYNCREQ resource table record. There are no reasons associated with a response of SCHEDULED.

Warning responses

A warning response indicates the API command was successfully processed, but a condition occurred that should be investigated. A REASON value is also returned that describes the condition. The following values represent a warning response:

NODATA A command that normally results in data being returned to the program was processed successfully, but there was no data to return. The reasons for a NODATA response are given with the commands that return it.

WARNING A command that normally results in data being returned to the program was processed successfully, but not all of the available data was returned. A typical reason for this response might be that the output area provided by the program was not large enough to hold all the data. The actual reasons for a WARNING response are given with the commands that return it.

Error responses

An error response indicates the API command was not successful. One or more REASON values are also returned that describe the error.

Note: Note that, except for the FAILED error response, these response codes usually indicate either an error in the user's API program (for example, failing to discard resources when they are no longer required), or an error with the CICSPlex SM environment (for example, a CMAS or MAS is not available).

The following values represent an error response:

BUSY A resource referred to by the command is currently being processed by another command. This situation can occur when a command that was previously issued with the NOWAIT option is processing a resource that is required by the current command. The reasons for a BUSY response are given with the commands that return it.

DUPE A resource referred to by the command already exists. The reasons for a DUPE response are given with the commands that return it.

ENVIRONERROR An environmental condition (such as short on storage) prevented the command from being processed. The reasons for an ENVIRONERROR response are given with the commands that return it.

FAILED An unexpected problem occurred during command processing. The reasons for a FAILED response are given with the commands that return it.

In the case of a FAILED EXCEPTION response, you should check the following sources for information related to the condition:

- EYULOG
- Job log
- AUXTRACE data set

INCOMPATIBLE	Two or more resources referred to by the command are incompatible. The reasons for an INCOMPATIBLE response are given with the commands that return it.
INUSE	A resource referred to by the command is in use and, therefore, cannot be discarded. The reasons for an INUSE response are given with the commands that return it.
INVALIDDATA	The command parameter list contains invalid data. The reason for an INVALIDDATA response is always the name of the parameter that contains invalid data. The reasons are given with the commands that return this response.
INVALIDPARM	<p>The command parameter list is invalid. There are a variety of situations that could result in an INVALIDPARM response. For example:</p> <p>Syntax error The syntax of an input parameter is incorrect (for example, a resource table name begins with a numeric character).</p> <p>Null parameter address An input parameter could not be found because the generated address for that parameter is 0.</p> <p>The reason for an INVALIDPARM response is always the name of the parameter that is invalid. The reasons are given with the commands that return this response.</p>
NOTAVAILABLE	A required CMAS or MAS resource is not available. The reasons for a NOTAVAILABLE response are given with the commands that return it.
NOTFOUND	A resource referred to by the command could not be found. The reasons for a NOTFOUND response are given with the commands that return it.
NOTPERMIT	The API request is not permitted by the external security manager (ESM) at your enterprise. The reasons for a NOTPERMIT response are given with the commands that return it.
SERVERGONE	The CMAS to which the processing thread was connected is no longer active. There are no reasons associated with a response of SERVERGONE.
TABLEERROR	An error was detected in a resource table record (either a result set record or a CICSplex SM definition record). The reasons for a TABLEERROR response are given with the commands that return it.

using RESPONSE and REASON

VERSIONINVL An invalid version of CICSplex SM was detected. The reasons for a VERSIONINVL response are given with the commands that return it.

Testing for RESPONSE and REASON

To evaluate the results of an API command, you simply code the RESPONSE and REASON options on the command and follow the command immediately with a test of the returned values. The RESPONSE and REASON options return numeric values. Different built-in functions are provided for converting and testing the numeric response and reason values in the command-level interface and the REXX run-time interface.

Using the command-level interface

When you are using the CICSplex SM command-level interface, you can use the EYUVALUE built-in function to convert and test the numeric RESPONSE and REASON values returned by an API command.

As an example, consider this API command:

```
EXEC CPSM CONNECT
      CONTEXT(WCONTEXT)
      SCOPE(WSCOPE)
      VERSION('0140')
      THREAD(WTHREAD)
      RESPONSE(WRESPONSE)
      REASON(WREASON)
      .
      .
```

To test for the RESPONSE value in each of the supported languages, you could code:

COBOL or PL/I:

```
IF WRESPONSE NOT = EYUVALUE(OK) GO TO NOCONNECT.
```

C/370:

```
if (WRESPONSE != EYUVALUE(OK)) { goto NOCONNECT; }
```

Assembler language:

```
CLC  WRESPONSE,EYUVALUE(OK)
BNE  NOCONNECT
```

which the built-in function changes to:

```
CLC  WRESPONSE,=F'1024'
```

You can use EYUVALUE in the same way to test for the REASON value, if the RESPONSE is one that returns a reason.

Using the REXX run-time interface

When you are using the REXX run-time interface, you can use the EYURESP and EYUREAS built-in functions to convert and test the numeric RESPONSE and REASON values returned by an API command.

As an example, consider this API command:

```

var = EYUAPI('CONNECT'           ,
            'CONTEXT('WCONTEXT')',
            'SCOPE('WSCOPE')'    ,
            'VERSION(0140)'      ,
            'THREAD(WTHREAD)'    ,
            'RESPONSE(WRESPONSE)',
            'REASON(WREASON)')
.
.

```

To test for the RESPONSE value, you could code:

```
If WRESPONSE <> EYURESP(OK) Then Signal NOCONNECT
```

to compare the numeric RESPONSE value returned in WRESPONSE with the numeric equivalent of OK.

Alternatively, you could code:

```
If EYURESP(WRESPONSE) <> "OK" Then Signal NOCONNECT
```

to convert the numeric RESPONSE value to its character equivalent first.

Note: The RESPONSE and REASON options report only run-time errors. Errors in interpreting an API command are reported in either the REXX RC variable or the variable assigned to a REXX function.

Retrieving FEEDBACK records

In addition to the specific values returned by a command's RESPONSE and REASON options, CICSPlex SM also provides diagnostic data in the form of FEEDBACK resource table records. This data can help you evaluate the results of an API command, especially if the command did not complete successfully.

Using the FEEDBACK command

You can retrieve diagnostic data about a previously issued API command by issuing the FEEDBACK command. The type of command for which you want diagnostic data affects how you specify the FEEDBACK command and where the data is placed:

A command that processed a result set

Use the RESULT option of the FEEDBACK command to retrieve data about the last command that processed a specific result set.

If the command that processed the result set returned a RESPONSE 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. You can use the FIRST, NEXT, and COUNT options of the FEEDBACK command to retrieve multiple FEEDBACK records.

The diagnostic data in a result set is available to the FEEDBACK command until another command processes the same result set. At that point, the data is replaced with FEEDBACK records for the subsequent command.

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

A command that did not process a result set

Use the FEEDBACK command without the RESULT option to retrieve data about the command issued immediately before FEEDBACK.

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.

Evaluating a FEEDBACK record

The diagnostic data for a CICSplex SM API command is presented in a FEEDBACK resource table record. The attributes of that resource table provide a variety of information about the completion status of an API command.

Note: This section provides general information about FEEDBACK records. The FEEDBACK resource table copy book that is supplied by CICSplex SM provides a detailed description of the contents and structure of a FEEDBACK record. You should refer to the *CICSplex SM Resource Tables Reference* or the supplied copy book when writing a program that uses the FEEDBACK command.

To identify which API operation the FEEDBACK record applies to, check the values in these fields:

COMMAND	A numeric code that identifies the command to which this FEEDBACK record applies. The API commands and their numeric equivalents are given in Table 7 on page 83.
OBJECT	The CICSplex SM object that the command was issued against.
OBJECT_ACT	The action that was being performed against the CICSplex SM object.
RSLTRECID	If the FEEDBACK record applies to a result set, the numeric ID of the result set record associated with this FEEDBACK record.

Table 7. Numeric codes and API commands

Numeric code	Mnemonic	Command
02	CANCEL	Cancel
03	CONNECT	Connect
04	COPY	Copy
05	CREATE	Create
06	DELETE	Delete
07	DISCARD	Discard
08	DISCONN	Disconnect
09	FETCH	Fetch
10	GET	Get
11	LOCATE	Locate
12	MARK	Mark
13	ORDER	Order
14	PERFSET	Perform Set
15	PERFOBJ	Perform Object
16	QUALIFY	Qualify
17	QUERY	Query
18	RECEIVE	Receive
19	REMOVE	Remove
20	FILTER	Specify Filter
21	UNMARK	Unmark
22	ADDRESS	Address
23	GETDEF	Getdef
24	LISTEN	Listen
25	REFRESH	Refresh
26	SET	Set
27	VIEW	Specify View
28	TERM	Terminate
29	TRANS	Translate
30	GROUP	Group by
31	UPDATE	Update

To determine what type of problem the FEEDBACK record describes, check the values in these fields:

ATTRDATAVAL Indicates whether attribute data is available for the command. Attribute data is included only if the command itself did not complete successfully.

If the ATTRDATAVAL value is Y, the FEEDBACK record identifies as many as five attributes (ATTR_NM1 through ATTR_NM5) that contributed to the error. Each attribute is identified by its name and its offset and relative number within the resource table record. The data type and length of each attribute is also included.

If the ATTRDATAVAL value is N, you can ignore the ATTR_ fields.

retrieving FEEDBACK records

- CEIBDATAVAL** Indicates whether CICS EIB data is available for the command. EIB data is included only if the command encountered a CICS error.
- If the CEIBDATAVAL value is Y, the FEEDBACK record includes the EIBFN, RESP, and RESP2 values as provided by CICS.
- If the CEIBDATAVAL value is N, you can ignore the CEIBFN, CEIBRESP, and CEIBRESP1 fields.
- ERRCODEVAL** Indicates whether a CICSplex SM error code is available for the command. An error code is included only if the command itself did not complete successfully.
- If the ERRCODEVAL value is Y, the FEEDBACK record includes a numeric ERROR_CODE value. Each resource table copy book includes a list of the error codes for that object and their meanings.
- If the ERRCODEVAL value is N, you can ignore the ERROR_CODE field, as well as the RESPONSE and REASON fields.

For some API operations that affect BAS resources, the FEEDBACK record may point to additional diagnostic data in an error result set. For more information about using the diagnostic data in error result sets, see “Additional processing for BAS” on page 86.

Availability of FEEDBACK records

In general, FEEDBACK records are produced for all API commands, whether they are successful or not. However, for some API commands and in some situations, FEEDBACK records are not produced because they would not provide useful diagnostic data.

FEEDBACK records are not available for these commands:

DISCONNECT and TERMINATE

When you disconnect an API processing thread from CICSplex SM, any remaining diagnostic data is discarded.

FEEDBACK

The FEEDBACK command cannot report on its own processing.

TBUILD and TPARSE

These REXX-specific commands issue a series of API commands internally and reuse the same feedback area. Therefore, the feedback area cannot represent the entire sequence of events.

FEEDBACK records are also not available in these situations:

- A command processes a result set and completes with a RESPONSE value of OK.
- A command is processed asynchronously (that is, you specify the NOWAIT option). The diagnostic data for asynchronous requests is returned in the ASYNCREQ notification resource table.

An example of FEEDBACK for a result set

As an example of how you can use FEEDBACK data, Figure 17 illustrates the results of issuing a SET command. In this case, SET was issued to modify the service status of CONNECT records in the result set referenced by TOKENC.

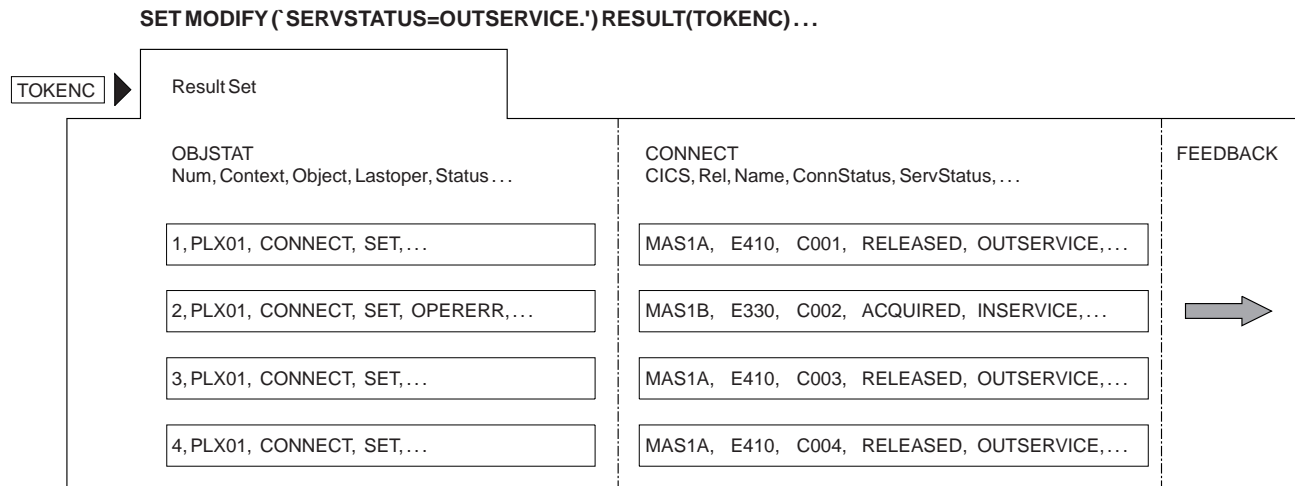


Figure 17. Using SET to modify result set records

One of the connections (C002 in MAS1B) was not successfully taken out of service by the SET command. The ServStatus field is still set to INSERVICE and there is a pointer to FEEDBACK data.

Figure 18 shows how you can use the FEEDBACK command to retrieve the FEEDBACK records associated with the result set referenced by TOKENC.

```
FEEDBACK RESULT(TOKENC) INTO(AREA5) ...
```

```
SET, N, Y, N, TABLEERROR, DATAERROR, ..., 16, 2, ..., CONNECT, ...
```

Figure 18. Using FEEDBACK to retrieve diagnostic data for a result set

The FEEDBACK record shown in Figure 18 reveals the cause of the problem. CICSplex SM returned RESPONSE and REASON values of TABLEERROR DATAERROR, which means the value associated with one or more resource table attributes is invalid. Furthermore, CICS responded to the SET request for this connection with RESP(16) RESP2(2). A check of the CICS response codes indicates that the attempt to take the connection out of service was invalid because the connection is currently acquired.

Note: The LASTOPER and STATUS attributes of the OBJSTAT resource table and some of the FEEDBACK attributes are actually binary fields (that is, they are represented by a bit being set on or off). For detailed information about the attribute values for a given resource table, refer to the *CICSplex SM Resource Tables Reference* or the supplied copy books.

Additional processing for BAS

For API operations that affect BAS resources, the diagnostic data in a FEEDBACK record may not be enough to fully describe an error condition. In these cases, the FEEDBACK record points to an error result set. An error result set is identified by the following fields:

- ERR_RESULT** A 4-byte token identifying an error result set.
- ERR_COUNT** The number of records in the error result set referenced by ERR_RESULT.
- ERR_OBJECT** The type of records in the error result set referenced by ERR_RESULT. This value is the 1- to 8-character name of a CICSplex SM resource table, and may be BINSTERR, BINCONRS, BINCONSC, or FEEDBACK.

Note: For details of the BINSTERR, BINCONRS, and BINCONSC resource tables, see the *CICSplex SM Resource Tables Reference*.

Evaluating error result set records

If the ERR_OBJECT field of the FEEDBACK record contains FEEDBACK, the error result set contains errors that arose when CICSplex SM attempted to update CICS resources. In response to the API command:

```
UPDATE RESULT(token) MODIFY(string)
```

CICSplex SM tries to update multiple CICS definition records in a result set according to the supplied modification string. For each CICS definition that could not be modified, an error record is created in the error result set. The RESPONSE and REASON values returned are TABLEERROR and DATAERROR.

The records are standard FEEDBACK records. To access the error result records, use the FEEDBACK command to retrieve diagnostic data about each of the CICS definitions in the ERR_RESULT result set. The ERR_COUNT value in the original FEEDBACK record for the UPDATE command indicates how many records are in the ERR_RESULT result set and therefore the number of times you should issue the FEEDBACK command against the ERR_RESULT result set.

Evaluating BINSTERR resource table records

If the ERR_OBJECT field of the FEEDBACK record contains BINSTERR, errors were encountered while CICS resources were being installed. In response to one of the following API commands:

```
PERFORM OBJECT ACTION(INSTALL)  
PERFORM SET ACTION(INSTALL)
```

CICSplex SM tries to install CICS resources in one or more active systems running CICS/ESA 4.1 or later. A BINSTERR record is created for each CICS resource that cannot be installed. The RESPONSE and REASON values returned are TABLEERROR and DATAERROR.

The BINSTERR records that you receive contain the following information:

- CMASNAME** The 1- to 8-character name of a CMAS that manages the specified CICSplex.

PLEXNAME	The 1- to 8-character name of the CICSplex to which the specified CICS system belongs.
CICSNAME	The 1- to 8-character name of the CICS system into which the resource could not be installed.
RESNAME	The name of the CICS resource that could not be installed.
RESVER	The version of the CICS definition that represents the resource being installed.
ERRCODE	A numeric CICSplex SM error code. See “BINSTERR” on page 102. The BINSTERR resource table copy book also contains a list of the error codes and their meanings.
CRESP1	The RESP value as returned by CICS.
CRESP2	The RESP2 value as returned by CICS.
CEIBFN	The EIBFN value as returned by CICS.

To access the error result set records, use the FETCH command to retrieve the BINSTERR records from the ERR_RESULT result set. The ERR_COUNT value in the FEEDBACK record for the PERFORM command indicates how many records are in the ERR_RESULT result set and therefore the number of times you should issue the FETCH command against the ERR_RESULT result set.

Evaluating BINCONRS resource table records

If the ERR_OBJECT field of the FEEDBACK record contains BINCONRS, inconsistent resource set errors were encountered when attempting to update or create the specified definition. In response to one of the following API commands:

```
CREATE OBJECT(basdef)
UPDATE OBJECT(basdef)
```

CICSplex SM tries to create or update one of the following Business Application Services definitions:

- RASGNDEF (resource assignment)
- RASINDSC (resource assignment in resource description)
- RESDESC (resource description)
- RESGROUP (resource group)
- RESINDSC (resource group in resource description)

A BINCONRS resource table record is created for each CICS definition that would cause an inconsistent set error. The RESPONSE and REASON values returned are TABLEERROR and DATAERROR.

The BINCONRS records that you receive contain the following information:

CMASNAME	The 1- to 8-character name of a CMAS that manages the specified CICSplex.
PLEXNAME	The 1- to 8-character name of the CICSplex to which the specified CICS system belongs.
CICSNAME	The 1- to 8-character name of the CICS system that experienced inconsistent resource set errors.
RESTYPE	The type of CICS resource.

ERROP	A numeric value that identifies the operation being performed when the error occurred (such as updating a RASGNDEF). See “BINCONRS” on page 101. The BINCONRS resource table copy book also contains a list of the ERROP values and their meanings.
CANDNAME	The name of the candidate resource
CANDVER	The version of the candidate resource
CANDRGRP	The group of the candidate resource
CANDRASG	The assignment of the candidate resource
CANDRDSC	The description of the candidate resource
CANDUSAGE	The candidate assignment usage
CANDSGRP	The candidate system group
CANDTYPE	The candidate system type
CANDASGOVR	The candidate assignment override
EXISTNAME	The name of the existing resource
EXISTVER	The version of the existing resource
EXISTRGRP	The group of the existing resource
EXISTRASG	The assignment of the existing resource
EXISTRDSC	The description of the existing resource
EXISTUSAGE	The existing assignment usage
EXISTSGRP	The existing system group
EXISTTYPE	The existing system type
EXISTASGOVR	The existing assignment override

To access the error result records, use the FETCH command to retrieve the BINCONRS records from the ERR_RESULT result set. The ERR_COUNT value in the FEEDBACK record for the CREATE or UPDATE command indicates how many records are in the ERR_RESULT result set and therefore the number of times you should issue the FETCH command against the ERR_RESULT result set.

Evaluating BINCONSC resource table records

If the ERR_OBJECT field contains BINCONSC, inconsistent scope errors were encountered while attempting to update or create the specified definition. In response to one of the following API commands:

```
CREATE OBJECT(basdef)
UPDATE OBJECT(basdef)
```

CICSplex SM tries to create or update one of the following Business Application Services definitions:

- RASGNDEF (resource assignment)
- RASINDSC (resource assignment in resource description)
- RESDESC (resource description)
- RESGROUP (resource group)
- RESINDSC (resource group in resource description)

A BINCONSC resource table record is created for each CICS definition that would cause an inconsistent scope error. The RESPONSE and REASON values returned are TABLEERROR and DATAERROR.

BINCONSC records contain the following information:

CMASNAME	The 1- to 8-character name of a CMAS that manages the specified CICSplex.
PLEXNAME	The 1- to 8-character name of the CICSplex to which the specified CICS system belongs.
CICSNAME	The 1- to 8-character name of the CICS system that experienced inconsistent scope errors.
ERROP	A numeric value that identifies the operation being performed when the error occurred (such as updating a RASGNDEF). See “BINCONSC” on page 101. The BINCONSC resource table copy book also contains a list of the ERROP values and their meanings.
ERRCODE	A numeric CICSplex SM error code. See “BINCONSC” on page 101. The BINCONSC resource table copy book contains a list of the error codes and their meanings.
TARGSCOPE	The name of the target scope
TARGRASG	The assignment for the target scope
TARGRDSC	The description for the target
RELSCOPE	The name of the related scope
RELRASG	The assignment for the related scope
RELRDSC	The description for the related scope

To access the error result records, use the FETCH command to retrieve the BINCONSC records from the ERR_RESULT result set. The ERR_COUNT value in the FEEDBACK record for the CREATE or UPDATE command indicates how many records are in the ERR_RESULT result set and therefore the number of times you should issue the FETCH command against the ERR_RESULT result set.

An example of a BAS error result set

As an example of how you can use the FEEDBACK data to obtain BAS error result set information, Figure 19 on page 90 illustrates the results of issuing a PERFORM OBJECT command. In this case, PERFORM OBJECT ACTION(INSTALL) was issued to install the CONNDEF definitions in the result set referenced by TOKENC.

PERFORM OBJECT ACTION (INSTALL) RESULT (TOKENC)

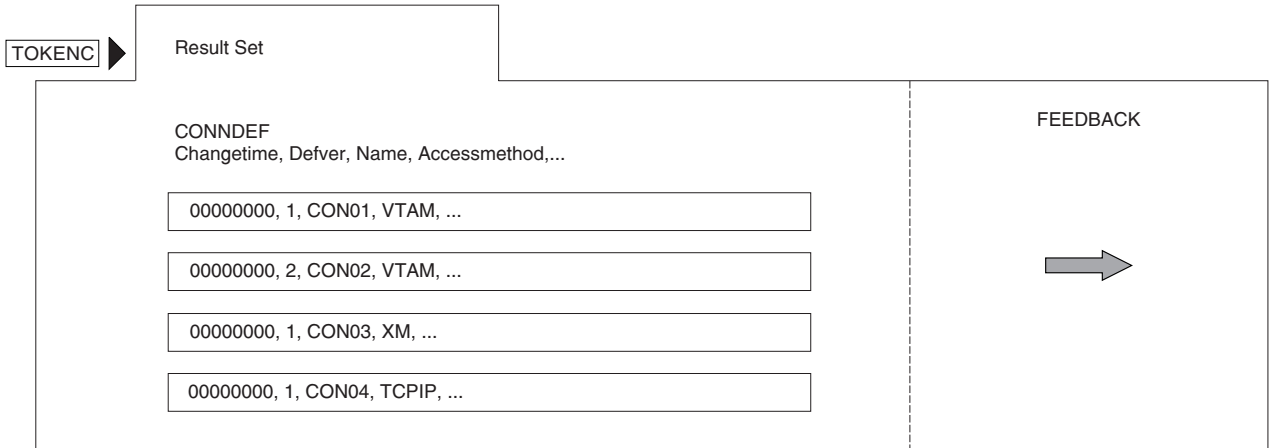


Figure 19. Using PERFORM OBJECT to install BAS definitions

One of the connection definitions (CON02,VTAM) was not successfully installed by the PERFORM OBJECT command. There is a pointer to the FEEDBACK data.

Figure 20 shows how you can use the FEEDBACK command to retrieve the FEEDBACK records associated with the result set referenced by TOKENC.

FEEDBACK RESULT (TOKENC) INTO (AREA5) ...

```
PERFORM OBJECT, N, Y, N, TABLEERROR, DATAERROR, ..., =>, 1, BINSTERR
```

Figure 20. Using FEEDBACK to retrieve diagnostic data for a result set

The FEEDBACK data shown in Figure 20 reveals the cause of the problem. CICSPlex SM returned RESPONSE and REASON values of TABLEERROR DATAERROR, which means that one or more connection definitions did not install successfully. Furthermore, the ERR_RESULT attribute points to an error result set which contains a single BINSTERR resource table record.

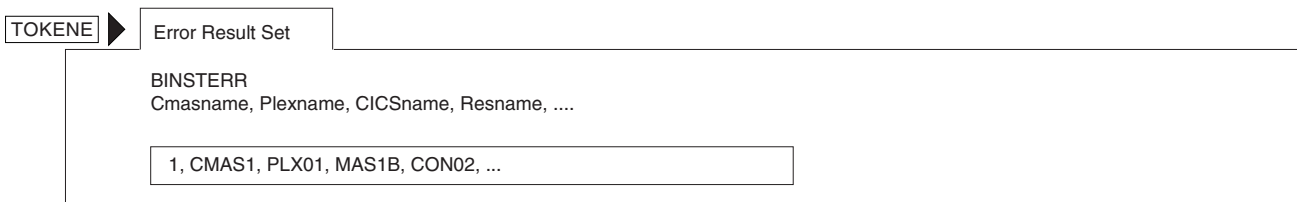


Figure 21. BINSTERR error result set

The BINSTERR error result set referenced by TOKENE, and shown in Figure 21 is accessed using a FETCH command.

FETCH RESULT (TOKENE) INTO (AREA6) ...

```
CMAS1, PLX01, MAS1B, CON02, 2, FORCENO, ...
```

Figure 22. Using FETCH to retrieve BINSTERR records

Figure 22 shows how you can use the FETCH command to retrieve the BINSTERR records associated with the error result set referenced by TOKENE.

Chapter 5. Writing a REXX program

This chapter describes how to use the REXX run-time interface to write an API program. It describes how to access the API through the REXX function package that is supplied with CICSplex SM, how to specify an API command, and how to process the data in a resource table record.

Accessing the API environment

The REXX run-time interface does not require any translation of API commands. The commands are interpreted by a REXX function package that is supplied by CICSplex SM.

Note: For instructions on installing the REXX function package, see *CICS Transaction Server for OS/390: Installation Guide*.

The REXX run-time interface consists of a single load module containing two entry points:

EYU9AR00 The function package
EYU9AR01 The host subcommand

The function package contains these functions:

EYUAPI() Passes an API command to CICSplex SM.
EYUINIT() Initializes the CICSplex SM API environment and allocates the necessary REXX resources.
EYUREAS() Translates the value returned by the REASON option of an API command.
EYURESP() Translates the value returned by the RESPONSE option of an API command.
EYUTERM() Terminates the CICSplex SM API environment and releases any allocated REXX resources.

For complete descriptions of these functions, see *CICSplex SM Application Programming Reference*.

In general, you access the CICSplex SM API environment through the supplied function package. The first call to CICSplex SM in your program must be an EYUINIT or EYUAPI function. EYUINIT is the primary means of initializing the API environment. However, if EYUINIT is not issued first, the EYUAPI function initializes the environment.

For example, sample program EYUxAPI1, which is distributed in the SEYUxxxx library, begins like this:

specifying an API command

```
Say 'Initializing API...'  
XX = EYUINIT()  
If XX <> 0 Then Signal UNEXPECTED  
Say 'Establishing connection...'  
XX = EYUAPI('CONNECT' ,  
           'CONTEXT('W_CONTEXT')' ,  
           'SCOPE('W_SCOPE')' ,  
           'VERSION(0140)' ,  
           'THREAD(W_THREAD)' ,  
           'RESPONSE(W_RESPONSE)' ,  
           'REASON(W_REASON)')  
If XX <> 0 Then Signal UNEXPECTED
```

In this example, the EYUINIT function is issued first to initialize the API environment. Then an EYUAPI function is used to issue the API CONNECT command.

Once you have issued an EYUINIT or EYUAPI function, you can:

- Issue any other CICSplex SM function.
- Access the host subcommand environment by issuing the REXX ADDRESS command.

Once the API environment is initialized, it exists until it is terminated, either by your program or by REXX. Therefore, the final call to CICSplex SM in your program should always be an EYUTERM function. If you do not issue EYUTERM, some REXX resources, such as storage, may remain allocated and REXX becomes responsible for releasing them.

For example, sample program EYUxAPI1 ends like this:

```
XX = EYUAPI('TERMINATE RESPONSE(W_RESPONSE) REASON(W_REASON)')  
XX = EYUTERM()
```

In this example, the EYUAPI function is used to issue an API TERMINATE command. Then EYUTERM is issued to terminate the API environment and release its allocated resources.

Using the EYUTERM function is always a good idea. However, 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 the EYUTERM function at the end of every program. Depending on the programming guidelines at your enterprise, the REXX resources that remain allocated can be reused by the next CICSplex SM API program that accesses the host subcommand environment.

Specifying an API command

When you write a program in REXX, you pass a character image of the command to be issued to the REXX function package supplied by CICSplex SM. The command string can include imbedded REXX variables, as appropriate. You can specify the command in one of two ways:

- Invoke the EYUAPI function with the name of the command as its parameter.
- Use the REXX ADDRESS command to pass subsequent statements to the function package.

Note: You can also use the REXX PARSE VALUE command to pass API commands to the function package. However, the processing overhead of PARSE VALUE is quite high. Furthermore, the EYUAPI function returns only a single character (0 or 1), so there is no need to parse its results. For these reasons, using PARSE VALUE is not recommended.

The following example shows a partial GET command as it would be issued using the EYUAPI function:

```
var = EYUAPI('GET OBJECT(LOCTRAN)...')
```

var is the variable assigned to receive the return code from the EYUAPI function.

The next example shows the same GET command being issued by the REXX ADDRESS command:

```
ADDRESS CPSM 'GET OBJECT(LOCTRAN)...'
```

When the data in a REXX variable is to be passed to the function package the text portion of the API command must be terminated, the REXX variable provided, and the rest of the API command completed. The following is an example of a complete GET command that demonstrates the imbedded use of REXX variables:

```
var = EYUAPI('GET OBJECT(LOCTRAN)' ,
            'RESULT(setvar) THREAD(THR1)' ,
            'RESPONSE(rspvar) REASON(reavar)')
```

In this example, the result set to receive the LOCTRAN objects, and the RESPONSE and REASON options are all specified as REXX variables.

Because of the way REXX handles variable substitution, you must keep in mind whether a variable is being used to send data to the API, receive data from the API, or both. The next example shows a CONNECT command where the USER and VERSION options send data to the API. The THREAD, RESPONSE, and REASON options all name variables to receive data from the API. Note that names of variables that receive data are specified as part of the command.

```
var = EYUAPI('CONNECT USER('userid') VERSION(0140)' ,
            'THREAD(thdtkn) RESPONSE(rspvar) REASON(reavar)')
```

In those cases where you want to access a resource table, special processing is required. An example of this is a FETCH command, which requires an INTO option to define where the resource table data should be placed for processing by your program. In REXX, you must specify the INTO option as the prefix of a stem variable to receive one or more resource table records. The zero entry of the stem variable indicates the number of records returned.

Accessing resource table data

Because of the way CICSplex SM supplies resource table data to REXX, two additional commands are provided as part of the REXX function package:

TPARSE Extracts individual resource table attributes from a record and places them into standard REXX variables. The resource table record itself can be supplied in any valid REXX variable, including a stem variable.

You can use TPARSE to break down and access the attribute data in a resource table record.

TBUILD Builds a CPSM Definition or CICS Definition resource table record from a set of variables that you supply. Each variable must contain an individual resource table attribute.

You can use TBUILD to build the resource table record for a definition that you want to create, update, or remove in the CICSplex SM data repository.

Note: TBUILD only uses 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).

The variables that represent the resource table attributes are created either by CICSplex SM, in the case of TPARSE, or by you, in the case of TBUILD. The variable names are formed by adding a prefix to the attribute name, like this:

```
prefix_fieldname
```

where:

prefix Is a text string that you supply. The maximum allowable length for a prefix is determined by REXX and the environment in which the program runs.

fieldname Is the name of an attribute in the resource table.

An underscore character (_) must be inserted between the prefix and the attribute name.

When a program written in REXX passes resource table records to the API, the format and layout of the record must be exactly as it is defined by CICSplex SM.

For complete descriptions of the TBUILD and TPARSE commands, see *CICSplex SM Application Programming Reference*.

Translating attribute values

The TBUILD and TPARSE commands use the TRANSLATE API command when processing certain resource table attributes. For example, EYUDA and CVDA values are maintained in a resource table record in their numeric form. By default, the TPARSE command converts these values into a displayable character form. TBUILD, on the other hand, converts any EYUDA or CVDA character values that you supply into their numeric equivalents.

However, if you use the ASIS option on these commands, attribute values are not converted. If you specify ASIS on the TPARSE command, you must also specify ASIS on the TBUILD command when you rebuild the record so that the API does not try to reconvert the values.

If you specify ASIS on the TPARSE command and then decide you want to convert the attribute values, you can use the TRANSLATE API command.

Processing **CHANGETIME** and **CREATETIME** attributes

The first 8 bytes of every CPSM Definition and CICS Definition resource table record contain an attribute called **CHANGETIME**, which reflects the date and time at which the record was last modified. CICS Definition records also include a **CREATETIME** attribute, which is the date and time at which the definition was created. The **CHANGETIME** and **CREATETIME** attributes are maintained internally by CICSplex SM; you should not attempt to modify these attribute values. When you update a resource table record, the **CHANGETIME** and **CREATETIME** values you pass to the **TBUILD** command must be the same values you received from **TPARSE**.

By default, the **TPARSE** command translates the **CHANGETIME** and **CREATETIME** values into displayable, character values. However, the character forms of these values cannot be passed back to **TBUILD**. So, if you plan to update a definition and then rebuild the resource table record, you should use the **ASIS** option on the **TPARSE** and **TBUILD** commands. When you use **ASIS**, the **CHANGETIME** and **CREATETIME** values appear as 16-byte hexadecimal values.

Chapter 6. REXX error handling

Translation errors

Errors that occur while REXX is trying to interpret a CICSPlex SM API command result in a REXX return code. If REXX cannot process a command string or function, the run-time interface sets the REXX return code in one of two places:

RC variable

When the ADDRESS CPSM command is used.

The return code value is one of the following:

- 0** The command was successfully processed.
- 8** The command contained syntax errors that prevented REXX from processing it. EYUARnnnn messages that describe the error are written to the destination defined on your system for IRXSAY WRITEERR output.
- 16** The command could not be processed because of some system failure (such as a lack of storage). REXX messages that describe the error may be produced.
- 3** The CICSPlex SM API environment is not available. This condition can occur if the function package is not properly installed. If the function package is installed, it could mean that you did not issue at least one EYUxxxx REXX function before invoking the ADDRESS CPSM command.

Function variable

When an EYUxxxx REXX function is used.

For most EYUxxxx functions, the return code value is one of the following:

- 0** The function was successfully processed.
- 1** The function failed. EYUARnnnn messages that describe the error are written to the destination defined on your system for IRXSAY WRITEERR output.

For the EYURESP and EYUREAS functions, the return code is either the numeric equivalent of the value being translated or -1, if the translation failed.

In general, if the REXX return code is anything other than:

- 0** From EYUAPI, EYUINIT, or EYUTERM

A valid RESPONSE or REASON value

From EYURESP or EYUREAS

the API command was not successfully interpreted by REXX and, therefore, was not passed to CICSPlex SM for processing. If a command is not processed, the RESPONSE and REASON values are not set and you do not need to check them.

If the return code is 0, the API command was interpreted by REXX and passed to CICSPlex SM. Note that a return code of 0 does not indicate whether the command was successfully processed by CICSPlex SM. To determine the results of an API command, refer to the RESPONSE and REASON values returned by the command.

Run-time errors

Errors that occur while CICSplex SM is trying to process an API command are reported by the RESPONSE and REASON values for the command. For more information, see “Using the RESPONSE and REASON options” on page 77.

TPARSE and TBUILD errors

The results of a TPARSE or TBUILD command are returned by the STATUS option, which is a required option on those commands. The STATUS option serves a similar purpose to the RESPONSE and REASON options on other API commands.

The STATUS option returns the REXX status value in character form as one of the following:

OK

The command completed processing successfully.

SYNTAX ERROR

The 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 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.

For more information about the EYUTRACE stem variable, see “EYU_TRACE data.”

Messages

Many of the error conditions you might encounter when using the REXX run-time interface are accompanied by messages that describe the error. These messages, which begin with the prefix EYUARnnnn, are written to the destination defined on your system for IRXSAY WRITEERR output. By default, such output goes to one of the following places:

- For a program running in TSO foreground, the output goes to the terminal.
 - For a program running in background, the output goes to the SYSTSPRT DD destination.
-

EYU_TRACE data

The run-time interface creates a REXX stem variable called EYU_TRACE anytime an error occurs that warrants tracing. Such conditions include:

- A STATUS of FAILURE from a TBUILD or TPARSE command
- A return code other than 0 from an EYUxxxx function.

The zero entry of the stem array indicates the number of trace records that were produced. Entries 1 through n contain the actual trace records.

If you are having problems with a REXX program or the run-time interface, IBM support may request the trace records from EYU_TRACE. CICSplex SM distributes a REXX EXEC that IBM support will ask you to include in your REXX program to format and print the EYU_TRACE records. The formatting routine is called EYU#TRCF and is distributed in the SEYUCLIB library. EYU#TRCF should be used only at the request of IBM support.

Appendix A. BINCONRS, BINCONSC, and BINSTERR error codes

This appendix contains the error codes in the BINCONRS, BINCONSC, and BINSTERR copy books. See “Retrieving FEEDBACK records” on page 81 for information on interpreting feedback error result sets containing these error codes.

BINCONRS

Table 8. BINCONRS error codes—ERROP field

Value	Code	Reason
01	ADDSYS	Add System to System Group
02	ADDTOGRP	Add Definition to Group
03	UPDINGRP	Update Definition in Group
04	ADDRASI	Add RASINDSC
05	ADDRESI	Add RESINDSC
06	UPDRASG	Update RASGNDEF
07	UPDRASI	Update RASINDSC
08	UPDRESI	Update RESDESC Install Scope
09	UPDRDSC	Update RESDESC

BINCONSC

Table 9. BINCONSC error codes—ERROP field

Value	Code	Reason
01	ADDRASI	Add RASINDSC
02	UPDRASG	Update RASGNDEF
03	UPDRASI	Update RASINDSC
04	UPDRDSC	Update RESDESC
05	ADDSYS	Add System to Group

Table 10. BINCONSC error codes—ERRCODE field

Value	Code	Reason
01	SAMESCP	Target/Related scopes are same
02	TRGINREL	Target Scope is in Related
03	RELINTRG	Related Scope is in Target
04	SYSNBOTH	CICSNAME in Target and Related
05	MULTREL	Multiple Systems in Related
06	RELNOSYS	Related System has no SYSID

BINSTERR*Table 11. BINSTERR error codes–ERRCODE field*

Value	Code	Reason
01	SYSSTATE	System inactive/not create capable
02	INSTNAUT	Install not authorized
03	DSCDNAUT	Discard not authorized
04	INSTFAIL	Install failure
05	DSCDFAIL	Install discard failure
06	INSTCPFL	Install Complete failure
07	INSTNCON	Install Connection failure
08	INSTSTAT	Install status failure
09	INSTNSUP	Install not supported
10	FORCENO	Resource Install negated
11	DSCRDERR	Discard failure
12	METHFAIL	MAS method failure
13	NOCREATE	System not create capable

Appendix B. Sample program listings

This appendix provides listings for the sample programs that are distributed with CICSplex SM. Each sample program is shown here in one of the languages in which it is distributed. For a list of the sample programs provided in each language and the libraries where they are distributed, see Table 6 on page 11.

Note: Additional sample CICSplex SM API programs are available on the World Wide Web at:

<http://www.software.ibm.com/ts/cicsplex/>

Sample program EYU#API1

Program EYU#API1 is written in REXX for the TSO environment.

EYUxAPI1 This program does the following:

- Establishes a connection to the API.
- Creates a result set containing all PROGRAM resource table records that do not begin with DFH, EYU, or IBM.
- Retrieves each record in the result set.
- Translates any CICS CVDA attributes into meaningful character values.
- Displays each record on the terminal, showing the program name, language, enable status, and CEDF status.
- Terminates the API connection.

Commands Used: CONNECT, FETCH, GET, TERMINATE, TRANSLATE

```

/* REXX */
/*****
/*
/* MODULE NAME = EYU#API1
/*
/* DESCRIPTIVE NAME = CPSM Sample API Program 1
/*                      (Sample REXX Version)
/*
/*                      5695-081
/*                      COPYRIGHT = NONE
/*
/* STATUS = %CP00
/*
/* FUNCTION =
/*
/* To provide an example of the use of the following EXEC CPSM
/* commands: CONNECT, GET, FETCH, TRANSLATE, TERMINATE.
/*
/* When invoked, the program depends upon the values held in the
/* W_CONTEXT and W_SCOPE declarations when establishing a
/* connection with CICSplex SM. They must take the following
/* values:
/*
/* W_CONTEXT = The name of a CMAS or CICSplex. Refer to the
/*              description of the EXEC CPSM CONNECT command
/*              for further information regarding the CONTEXT
/*              option.
/*
/* W_SCOPE   = The name of a CICSplex, CICS system, or CICS
/*              system group within the CICSplex. Refer to the
/*              description of the EXEC CPSM CONNECT command
/*              for further information regarding the SCOPE
/*              option.
/*
/* This sample requires no parameters at invocation time.
/*
/* The sample establishes an API connection and issues a GET
/* command to create a result set containing program resource
/* table records which match the criteria.
/*
/* Using the FETCH command each record in the result set is
/* retrieved. Once retrieved the TRANSLATE command is used to
/* convert those attributes of each record which are EYUDA or
/* CVDA values into meaningful character representations. A
/* record is then displayed on the terminal showing the program
/* name, language, program status, and CEDF status.
/*
/* Finally, the API connection is terminated.
/*
/*-----
/*NOTES :
/* DEPENDENCIES = S/390, TSO
/* RESTRICTIONS = None
/* REGISTER CONVENTIONS =
/* MODULE TYPE   = Executable
/* PROCESSOR     = REXX
/* ATTRIBUTES    = Read only, Serially Reusable
/*

```

sample program EYU#API1

```
/*-----*/
/*                                          */
/*ENTRY POINT = EYU#API1                    */
/*                                          */
/*  PURPOSE = All Functions                 */
/*                                          */
/*  LINKAGE = From TSO as a REXX EXEC.     */
/*                                          */
/*  INPUT   = None.                        */
/*                                          */
/*-----*/
/*                                          */
Address 'TSO'
Parse Value 0 0 With W_RESPONSE W_REASON .
/*-----*/
/*  CHANGE W_CONTEXT AND W_SCOPE TO MATCH YOUR INSTALLATION */
/*-----*/
W_CONTEXT = 'RTGA'
W_SCOPE = 'RTGA'
/*-----*/
/*  OBTAIN A CPSM API CONNECTION.          */
/*                                          */
/*  THE API WILL RETURN A TOKEN IDENTIFYING THE THREAD IN */
/*  VARIABLE W_THREAD.                    */
/*-----*/
Say 'Initializing API...'
XX = EYUINIT()
If XX <> 0 Then Signal UNEXPECTED
Say 'Establishing connection...'
XX = EYUAPI('CONNECT' ,
           'CONTEXT('W_CONTEXT')' ,
           'SCOPE('W_SCOPE')' ,
           'VERSION(0140)' ,
           'THREAD(W_THREAD)' ,
           'RESPONSE(W_RESPONSE)' ,
           'REASON(W_REASON)')
If XX <> 0 Then Signal UNEXPECTED
If W_RESPONSE <> EYURESP(OK) Then Signal NO_CONNECT
/*-----*/
/*  GET THE PROGRAM RESOURCE TABLE.      */
/*                                          */
/*  CREATE A RESULT SET CONTAINING ENTRIES FOR ALL PROGRAMS */
/*  WITH NAMES NOT BEGINNING DFH, EYU or IBM.                */
/*  THE NUMBER OF ENTRIES MEETING THE CRITERIA IS RETURNED IN */
/*  VARIABLE W_RECcnt.                                       */
/*-----*/
Say 'Get the PROGRAM resource table...'
W_CRITERIA = 'NOT (PROGRAM=DFH* OR PROGRAM=EYU* OR PROGRAM=IBM*).'
W_CRITERIALEN = 'LENGTH'(W_CRITERIA)
XX = EYUAPI('GET OBJECT(PROGRAM)' ,
           'CRITERIA(W_CRITERIA)' ,
           'LENGTH('W_CRITERIALEN)')' ,
           'COUNT(W_RECcnt)' ,
           'RESULT(W_RESULT)' ,
           'THREAD(W_THREAD)' ,
           'RESPONSE(W_RESPONSE)' ,
           'REASON(W_REASON)')
If XX <> 0 Then Signal UNEXPECTED
If W_RESPONSE <> EYURESP(OK) Then Signal NO_GET
```

```

/*-----*/
/*  RETRIEVE INFORMATION ABOUT EACH PROGRAM.          */
/*-----*/
/*  FETCH EACH ENTRY AND USE TPARSE TO OBTAIN EACH ATTRIBUTE.  */
/*  DISPLAY DETAILS OF EACH PROGRAM TO THE USER.          */
/*-----*/
Say 'Fetching' W_RECcnt 'PROGRAM entries...'
Say 'Program Language      Status      CEDF Status'
W_INTO_OBJECTLEN = 136          /* LENGTH OF PROGRAM TABLE */
Do III = 1 To W_RECcnt
  XX = EYUAPI('FETCH INTO(W_INTO_OBJECT)' ,
             'LENGTH(W_INTO_OBJECTLEN)' ,
             'RESULT(W_RESULT)' ,
             'THREAD(W_THREAD)' ,
             'RESPONSE(W_RESPONSE)' ,
             'REASON(W_REASON)')

  If XX <> 0 Then Signal UNEXPECTED
  If W_RESPONSE <> EYURESP(OK) Then Signal NO_FETCH
  XX = EYUAPI('TPARSE OBJECT(PROGRAM)' ,
             'PREFIX(PGM)' ,
             'STATUS(W_RESPONSE)' ,
             'VAR(W_INTO_OBJECT.1)' ,
             'THREAD(W_THREAD)')

  If W_RESPONSE <> 'OK' Then Signal UNEXPECTED
  W_TEXT = PGM_PROGRAM
  W_TEXT = 'OVERLAY'(PGM_LANGUAGE,W_TEXT,10)
  W_TEXT = 'OVERLAY'(PGM_STATUS,W_TEXT,23)
  W_TEXT = 'OVERLAY'(PGM_CEDFSTATUS,W_TEXT,36)
  Say W_TEXT
End III
Signal ENDIT
/*-----*/
/*  PROCESSING FOR API FAILURES.                      */
/*-----*/
UNEXPECTED:
  W_MSG_TEXT = 'UNEXPECTED ERROR.'
  Signal SCRNLG
NO_CONNECT:
  W_MSG_TEXT = 'ERROR CONNECTING TO API.'
  Signal SCRNLG
NO_GET:
  W_MSG_TEXT = 'ERROR GETTING RESOURCE TABLE.'
  Signal SCRNLG
NO_FETCH:
  W_MSG_TEXT = 'ERROR FETCHING RESULT SET.'
  Signal SCRNLG
SCRNLG:
  Say W_MSG_TEXT
  Say 'RESPONSE='||W_RESPONSE ,
      'REASON='||W_REASON 'RESULT='XX
ENDIT:
/*-----*/
/*  TERMINATE API CONNECTION.                          */
/*-----*/
XX = EYUAPI('TERMINATE RESPONSE(W_RESPONSE) REASON(W_REASON)')
XX = EYUTERM()
Exit

```

|
|
|
|
|

The C/370, COBOL and PL/1 versions of EYUxAPI1 are written for the CICS environment and can be converted to run the MVS/ESA batch environment by commenting the EXEC CICS SEND commands, and uncommenting the preceding language specific output statements.

Sample program EYUCAPI2

Program EYUCAPI2 is written in C/370 for the CICS ENVIRONMENT.

EYUxAPI2 This program does the following:

- Establishes a connection to the API.
- Defines a filter to identify PROGRAM resource table records with a language attribute of Assembler.
- Creates a result set containing all PROGRAM resource table records that do not begin with DFH, EYU, or IBM.
- Marks those records in the result set that match the specified filter (LANGUAGE=ASSEMBLER).
- Copies the marked records to a new result set.
- Deletes the marked records from the original result set.
- For each result set (LANGUAGE=ASSEMBLER and LANGUAGE≠ASSEMBLER):
 - Retrieves each record.
 - Translates any CICS CVDA attributes.
 - Displays each record on the terminal.
- Terminates the API connection.

Commands Used: CONNECT, COPY, DELETE, FETCH, GET, LOCATE, MARK, SPECIFY FILTER, TERMINATE, TRANSLATE


```

/*****/
/*
/* MODULE NAME = EYUCAPI2
/*
/* DESCRIPTIVE NAME = CPSM Sample API Program 2
/*                      (Sample C Version)
/*
/*      5695-081
/*      COPYRIGHT = NONE
/*
/* STATUS = %CP00
/*
/* FUNCTION =
/*
/* To provide an example of the use of the following EXEC CPSM
/* commands: CONNECT, SPECIFY FILTER, GET, MARK, COPY, DELETE,
/* LOCATE, FETCH, TRANSLATE, TERMINATE.
/*
/* When invoked, the program depends upon the values held in the
/* W_CONTEXT and W_SCOPE declarations when establishing a
/* connection with CICSplex SM. They must take the following
/* values:
/*
/* W_CONTEXT = The name of a CMAS or CICSplex. Refer to the
/*              description of the EXEC CPSM CONNECT command
/*              for further information regarding the CONTEXT
/*              option.
/*
/* W_SCOPE   = The name of a CICSplex, CICS system, or CICS
/*              system group within the CICSplex. Refer to the
/*              description of the EXEC CPSM CONNECT command
/*              for further information regarding the SCOPE
/*              option.
/*
/* This sample requires no parameters at invocation time.
/*
/* The sample establishes an API connection and issues a SPECIFY
/* FILTER command to create a filter which will match only
/* specific program resource table records. The filter is used
/* later in the program by the MARK command.
/*
/* A GET command is issued to create a result set containing
/* program resource table records which match the criteria. The
/* result set is then used by the MARK command to flag records
/* meeting the previous filter specification. The marked records
/* are then COPYed to a new result set, and then DELETED from
/* the original result set. After this sequence of commands we
/* have two results sets; one containing records which did not
/* meet the filter specification (that is, records where the
/* LANGUAGE is not ASSEMBLER), and one containing records
/* which did match the filter (that is, records where the
/* LANGUAGE is ASSEMBLER).
/*
/* Taking each of the two results sets in turn a LOCATE command
/* is used to ensure we start at the top of the result set
/* before a FETCH command is used to retrieve each record in
/* the result set. Once retrieved the TRANSLATE command is used
/* to convert those attributes of each record which are EYUDA
/* or CVDA values into meaningful character representations. A
/* record is then displayed on the terminal showing the program
/* name, language, program status, and CEDF status.
/*
/* Finally, the API connection is terminated.
/*

```

sample program EYUCAPI2

```

* -----*
/*NOTES : */
/* DEPENDENCIES = S/390, CICS */
/* RESTRICTIONS = None */
/* REGISTER CONVENTIONS = */
/* MODULE TYPE = Executable */
/* PROCESSOR = C */
/* ATTRIBUTES = Read only, Serially Reusable */
/*
/*-----*/
/*
/*ENTRY POINT = EYUCAPI2 */
/*
/* PURPOSE = All Functions */
/*
/* LINKAGE = From CICS either with EXEC CICS LINK or as a CICS */
/* transaction. */
/*
/* INPUT = None. */
/*-----*/
/*
#include <PROGRAM>
void main()
{
/*-----*/
/* CHANGE W_CONTEXT AND W_SCOPE TO MATCH YOUR INSTALLATION */
/*-----*/
char *W_CONTEXT = "RTGA ";
char *W_SCOPE = "RTGA ";
int W_RESPONSE;
int W_REASON;
int W_THREAD;
char *W_CRITERIA;
int W_CRITERIALEN;
int W_FILTER_TOKEN;
int W_RESULT = 0;
int W_COUNT;
int W_RESULT2 = 0;
int W_COUNT2;
int III;
int JJJ;
int W_RESULT_TOK;
int W_RECcnt;
PROGRAM W_INT0_OBJECT;
int W_INT0_OBJECTLEN;
char W_TRANSCVDA??(12??);
char W_TEXT??(81??);
char W_MSG_TEXT??(81??);
W_TEXT??(80??) = 0x13;
W_MSG_TEXT??(80??) = 0x13;
/*-----*/
/* OBTAIN A CPSM API CONNECTION. */
/*
/* THE API WILL RETURN A TOKEN IDENTIFYING THE THREAD IN */
/* VARIABLE W_THREAD. */
/*-----*/
strcpy(W_TEXT,"Establishing connection...");
/* printf("Establishing connection...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) ERASE;
EXEC CPSM CONNECT
CONTEXT(W_CONTEXT)
SCOPE(W_SCOPE)
VERSION("0140")
THREAD(W_THREAD)
RESPONSE(W_RESPONSE)
REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_CONNECT; }

```

```

/*-----*/
/*   CREATE A FILTER.                                     */
/*                                                     */
/*   CREATE A FILTER WHICH WILL MATCH ONLY THOSE PROGRAMS WITH */
/*   A LANGUAGE OF ASSEMBLER.                             */
/*   THE FILTER WILL BE USED IN A SUBSEQUENT MARK COMMAND.  */
/*-----*/
strcpy(W_TEXT,"Create a filter... ");
/* printf("Create a filter...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
W_CRITERIA = "LANGUAGE=ASSEMBLER.";
W_CRITERIALEN = strlen(W_CRITERIA);
EXEC CPSM SPECIFY FILTER(W_FILTER_TOKEN)
        CRITERIA(W_CRITERIA)
        LENGTH(W_CRITERIALEN)
        OBJECT("PROGRAM ")
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_FILTER; }

/*-----*/
/*   GET THE PROGRAM RESOURCE TABLE.                   */
/*                                                     */
/*   CREATE A RESULT SET CONTAINING ENTRIES FOR ALL PROGRAMS */
/*   WITH NAMES NOT BEGINNING DFH, EYU OR IBM.           */
/*   THE NUMBER OF ENTRIES MEETING THE CRITERIA IS RETURNED IN */
/*   VARIABLE W_COUNT.                                   */
/*-----*/
strcpy(W_TEXT,"Get the PROGRAM resource table...");
/* printf("Get the PROGRAM resource table...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
W_CRITERIA = "NOT (PROGRAM=DFH* OR PROGRAM=EYU* OR PROGRAM=IBM*).";
W_CRITERIALEN = strlen(W_CRITERIA);
EXEC CPSM GET OBJECT("PROGRAM ")
        CRITERIA(W_CRITERIA)
        LENGTH(W_CRITERIALEN)
        COUNT(W_COUNT)
        RESULT(W_RESULT)
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_GET; }
sprintf(W_TEXT,"Total number of entries: %d", W_COUNT);
/* printf(W_TEXT); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;

/*-----*/
/*   MARK SELECTED PROGRAM ENTRIES.                     */
/*                                                     */
/*   USING THE FILTER WE MARK THOSE ENTRIES IN THE RESULT SET */
/*   WHICH MEET THE FILTER SPECIFICATION IE. THOSE ENTRIES WITH */
/*   A LANGUAGE OF ASSEMBLER.                             */
/*-----*/
strcpy(W_TEXT,"Mark LANGUAGE=ASSEMBLER entries...");
/* printf("Mark LANGUAGE=ASSEMBLER entries...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
EXEC CPSM MARK FILTER(W_FILTER_TOKEN)
        RESULT(W_RESULT)
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_MARK; }

```

sample program EYUCAPI2

```
/*-----*/
/* COPY MARKED ENTRIES TO ANOTHER RESULT SET. */
/* */
/* HAVING MARKED ENTRIES IN THE RESULT SET WE CAN COPY THEM */
/* TO A NEW RESULT SET. */
/* AFTER THIS COMMAND WE WILL HAVE TWO RESULT SETS. ONE */
/* CONTAINING ALL THE PROGRAM ENTRIES, AND THE OTHER CONTAINING */
/* JUST THOSE ENTRIES WITH A LANGUAGE OF ASSEMBLER. */
/*-----*/
strcpy(W_TEXT,"Copy marked entries... ");
/* printf("Copy marked entries...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
EXEC CPSM COPY FROM(W_RESULT)
      TO(W_RESULT2)
      MARKED
      COUNT(W_COUNT2)
      THREAD(W_THREAD)
      RESPONSE(W_RESPONSE)
      REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_COPY; }
sprintf(W_TEXT,"Number of entries copied: %d", W_COUNT2);
/* printf(W_TEXT); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
/*-----*/
/* DELETE MARKED ENTRIES FROM RESULT SET. */
/* */
/* WE CAN NOW DELETE THE MARKED ENTRIES FROM THE ORIGINAL */
/* RESULT SET. */
/* AFTER THIS COMMAND WE HAVE TWO RESULT SETS. ONE RESULT SET */
/* CONTAINING ENTRIES WITH LANGUAGE NOT ASSEMBLER, AND THE */
/* OTHER CONTAINING ENTRIES WITH A LANGUAGE OF ASSEMBLER. */
/*-----*/
strcpy(W_TEXT,"Delete marked entries... ");
/* printf("Delete marked entries...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
EXEC CPSM DELETE MARKED
      COUNT(W_COUNT)
      RESULT(W_RESULT)
      THREAD(W_THREAD)
      RESPONSE(W_RESPONSE)
      REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_DELETE; }
sprintf(W_TEXT,"Number of entries remaining: %d", W_COUNT);
/* printf(W_TEXT); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
```

```

/*-----*/
/*  RETRIEVE INFORMATION ABOUT EACH PROGRAM.          */
/*                                                    */
/*  FETCH EACH ENTRY, USE INCLUDED STRUCTURE TO OBTAIN EACH */
/*  ATTRIBUTE AND USE TRANSLATE TO CONVERT CICS CVDAS.    */
/*  DISPLAY DETAILS OF EACH PROGRAM TO THE USER.        */
/*-----*/
W_INT0_OBJECTLEN = PROGRAM_TBL_LEN;
for (JJJ = 1; JJJ <= 2; JJJ++)
{
  if (JJJ == 1)
  {
    sprintf(W_TEXT,"Fetching %d non-ASSEMBLER PROGRAM entries...\n",
            W_COUNT);
    W_RESULT_TOK = W_RESULT;
    W_RECcnt = W_COUNT;
  }
  else
  {
    sprintf(W_TEXT,"Fetching %d ASSEMBLER PROGRAM entries...\n",
            W_COUNT2);
    W_RESULT_TOK = W_RESULT2;
    W_RECcnt = W_COUNT2;
  }
  /* printf(W_TEXT); */
  EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
  EXEC CPSM LOCATE TOP
        RESULT(W_RESULT_TOK)
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
  if (W_RESPONSE != EYUVALUE(OK)) { goto NO_LOCATE; }
  strcpy(W_TEXT,"Program Language Status CEDF Status");
  /* printf("Program Language Status CEDF Status\n"); */
  EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
  for (III = 1; III <= W_RECcnt; III++)
  {
    EXEC CPSM FETCH INTO(&W_INT0_OBJECT)
          LENGTH(W_INT0_OBJECTLEN)
          RESULT(W_RESULT_TOK)
          THREAD(W_THREAD)
          RESPONSE(W_RESPONSE)
          REASON(W_REASON) ;
    if (W_RESPONSE != EYUVALUE(OK)) { goto NO_FETCH; }
    memcpy(W_TEXT,W_INT0_OBJECT.PROGRAM,8);
    EXEC CPSM TRANSLATE OBJECT("PROGRAM ")
          ATTRIBUTE("LANGUAGE ")
          FROMCV(W_INT0_OBJECT.LANGUAGE)
          TOCHAR(W_TRANSCVDA)
          THREAD(W_THREAD)
          RESPONSE(W_RESPONSE)
          REASON(W_REASON) ;
    if (W_RESPONSE != EYUVALUE(OK)) { goto NO_TRANSLATE; }
    memcpy(W_TEXT+9,W_TRANSCVDA,12);
    EXEC CPSM TRANSLATE OBJECT("PROGRAM ")
          ATTRIBUTE("STATUS ")
          FROMCV(W_INT0_OBJECT.STATUS)
          TOCHAR(W_TRANSCVDA)
          THREAD(W_THREAD)
          RESPONSE(W_RESPONSE)
          REASON(W_REASON) ;
  }
}

```

sample program EYUCAPI2

```
        if (W_RESPONSE != EYUVALUE(OK)) { goto NO_TRANSLATE; }
        memcpy(W_TEXT+22,W_TRANSCVDA,12);
        EXEC CPSM TRANSLATE OBJECT("PROGRAM ")
            ATTRIBUTE("CEDFSTATUS ")
            FROMCV(W_INTTO_OBJECT.CEDFSTATUS)
            TOCHAR(W_TRANSCVDA)
            THREAD(W_THREAD)
            RESPONSE(W_RESPONSE)
            REASON(W_REASON) ;
        if (W_RESPONSE != EYUVALUE(OK)) { goto NO_TRANSLATE; }
        memcpy(W_TEXT+35,W_TRANSCVDA,12);
        /* printf("%s\n",W_TEXT); */
        EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
    }
}
goto ENDIT;
/*-----*/
/*   PROCESSING FOR API FAILURES.                               */
/*-----*/
NO_CONNECT:
    strcpy(W_MSG_TEXT,"ERROR CONNECTING TO API.\n");
    goto SCRNL0G;
NO_FILTER:
    strcpy(W_MSG_TEXT,"ERROR CREATING FILTER.\n");
    goto SCRNL0G;
NO_GET:
    strcpy(W_MSG_TEXT,"ERROR GETTING RESOURCE TABLE.\n");
    goto SCRNL0G;
NO_MARK:
    strcpy(W_MSG_TEXT,"ERROR MARKING RESULT SET.\n");
    goto SCRNL0G;
NO_COPY:
    strcpy(W_MSG_TEXT,"ERROR COPYING RESULT SET.\n");
    goto SCRNL0G;
NO_DELETE:
    strcpy(W_MSG_TEXT,"ERROR DELETING FROM RESULT SET.\n");
    goto SCRNL0G;
NO_LOCATE:
    strcpy(W_MSG_TEXT,"ERROR LOCATING TO TOP OF RESULT SET.\n");
    goto SCRNL0G;
NO_FETCH:
    strcpy(W_MSG_TEXT,"ERROR FETCHING RESULT SET.\n");
    goto SCRNL0G;
NO_TRANSLATE:
    strcpy(W_MSG_TEXT,"ERROR TRANSLATING ATTRIBUTE\n");
    goto SCRNL0G;
SCRNL0G:
    /* printf(W_MSG_TEXT); */
    EXEC CICS SEND FROM(W_MSG_TEXT) LENGTH(81) WAIT;
    sprintf(W_MSG_TEXT,"RESPONSE=%d REASON=%d\n",W_RESPONSE,W_REASON);
    /* printf(W_MSG_TEXT); */
    EXEC CICS SEND FROM(W_MSG_TEXT) LENGTH(81) WAIT;
ENDIT:
/*-----*/
/*   TERMINATE API CONNECTION.                               */
/*-----*/
EXEC CPSM TERMINATE RESPONSE(W_RESPONSE) REASON(W_REASON);
EXEC CICS RETURN;
}
```

The C/370, COBOL and PL/1 versions of EYUxAPI2 are written for the CICS environment and can be converted to run the MVS/ESA batch environment by commenting the EXEC CICS SEND commands, and uncommenting the preceding language specific output statements.

Sample program EYUAAPI3

Program EYUAAPI3 is written in Assembler H for the MVS/ESA batch environment.

EYUxAPI3 This program does the following:

- Establishes a connection to the API with the context set to an existing CICSplex.
- Verifies that a proposed new CICSplex name is not already defined to CICSplex SM as a CICSplex, CMAS, CICS system, or CICS system group.
- Creates a result set containing the CPLEXDEF resource table record for the existing CICSplex definition and retrieves that record.
- Creates a new CPLEXDEF resource table record using the existing record as a model.
- Creates a result set containing the CICSplex resource table records associated with the existing CICSplex and retrieves those records.
- Creates new CICSplex resource table records using the existing records as models.
- Sequentially retrieves all the resource table records associated with the existing CICSplex, including CICS systems, CICS system groups, workload management definitions, real-time analysis definitions, and resource monitoring definitions.
- Creates all the necessary resource table records for the new CICSplex using the existing records as models.
- If an error occurs before all the necessary resource table records are created, removes the new CICSplex definition.
- Disconnects the API processing thread.

Commands Used: CONNECT, CREATE, DISCARD, DISCONNECT, FETCH, GET, PERFORM OBJECT, QUALIFY, QUERY, REMOVE

sample program EYUAAPI3

```
*
EYUAAPI3 TITLE 'EYUAAPI3 - CPSM SAMPLE API PROGRAM 3 - ASSEMBLER'
*****
*
* MODULE NAME = EYUAAPI3
*
* DESCRIPTIVE NAME = API sample program 3 ASSEMBLER Version
*
*      5695-081
*      COPYRIGHT = NONE
*
* STATUS = %CP00
*
* FUNCTION =
*
*   To mirror an existing PLEX to a new PLEX.
*
*   When invoked, the program depends upon the values held in the
*   OLDPLEX, NEWPLEX, and MPCMAS variables. They must be set to
*   the following values:
*
*   OLDPLEX   = The name of an existing PLEX that will be mirrored.
*
*   NEWPLEX   = The name that will be given to the new PLEX.
*
*   MPCMAS    = The maintenance point CMAS of the OLDPLEX. This
*               will also be the MP for the NEWPLEX.
*
*   This sample requires no parameters at invocation time.
*
*   The sample processes as follows:
*
*   - a CONNECTION is established to CPSM, with the CONTEXT and
*     SCOPE of the OLDPLEX.
*
*   - since a PLEX can be either a CONTEXT or SCOPE, we verify
*     that the NEWPLEX is not already a valid CONTEXT (i.e, an
*     existing CICSplex or CMAS) or SCOPE in the OLDPLEX (i.e,
*     an existing CICS system or CICS system group).
*
*   - we GET the CPLEXDEF record for the OLDPLEX, and use this as
*     a module to CREATE the NEWPLEX.
*
*   - we GET the CICSplex records for the OLDPLEX, and use these
*     to add the CMASs in the OLDPLEX to the NEWPLEX.
*
*   - using a list that contains all possible CICSplex definitions,
*     we GET and FETCH the records from the OLDPLEX, and CREATE
*     them in the NEWPLEX.
*
*   - we then DISCONNECT from CPSM.
*
*
```



```

* -----*
*
* NOTES :
* DEPENDENCIES = S/370
* RESTRICTIONS = None
* REGISTER CONVENTIONS =
*   R0      Workarea / external call parameter pointer
*   R1      Workarea / external call parameter pointer
*   R2      Resource Table record pointer
*   R3      Loop counter
*   R4      List pointer
*   R5      Loop counter
*   R6      Unused
*   R7      Unused
*   R8      Unused
*   R9      Subroutine linkage
*   R10     Subroutine linkage
*   R11     Base register
*   R12     Base register
*   R13     Workarea pointer
*   R14     External call linkage
*   R15     External call linkage
*
* MODULE TYPE = Executable
* PROCESSOR   = Assembler
* ATTRIBUTES  = Read only, Serially Reusable
*              AMODE(31), RMODE(ANY)
*
* -----*
* ENTRY POINT = EYUAAPI3
*
* PURPOSE = All Functions
*
* LINKAGE = Executed as a batch program.
*
* INPUT   = None
*
* OUTPUT  = File for messages.
*          DDNAME = SYSPRINT
*          DSORG  = PS
*          RECFM  = FB
*          LRECL  = 80
*          BLKSIZE = as desired (a multiple of 80)
*
* -----*
EJECT
EYUAAPI3 CSECT
STM  R14,R12,12(R13)
LR   R12,R15
USING EYUAAPI3,R12
* -----*
* GETMAIN working storage and set up SA chain.
* -----*
GETMAIN R,LV=WORKLEN
ST  R13,4(,1)
ST  R1,8(,R13)
L   R1,24(,R13)
L   R13,8(,R13)
USING SAVEAREA,R13

```

sample program EYUAAPI3

```

*-----*
*       Preset return code to error - will change to 0 if all ok.   *
*-----*
MVC  RETCODE,=F'8'
*-----*
*       OPEN file for error messages.                               *
*-----*
OPEN  (SYSPRINT,OUTPUT)
*-----*
*       Specify variables: OLDPLEX, NEWPLEX, MPCMAS                 *
*-----*
*       Insure that the values specified are valid NAME type (i.e,  *
*       valid member name) or following code will fail.           *
*-----*
MVC  OLDPLEX,=CL8'plexold'    *** SPECIFY AS DESIRED ***
MVC  NEWPLEX,=CL8'plexnew'    *** SPECIFY AS DESIRED ***
MVC  MPCMAS,=CL8'mpcmas'     *** SPECIFY AS DESIRED ***
*-----*
*       Connect to CPSM API via OLDPLEX.                            *
*-----*
MVC  CONTEXT,OLDPLEX
EXEC  CPSM CONNECT                                X
      CONTEXT(CONTEXT)                          X
      VERSION(=CL4'0130')                        X
      THREAD(THREAD)                             X
      RESPONSE(RESPONSE)                         X
      REASON(REASON)
CLC  RESPONSE,EYUVALUE(OK)    RESPONSE OK?
BNE  ERRCON                  No - msgs and out
*-----*
*       Verify that the desired NEWPLEX name is not already a     *
*       PLEX or CMAS. We do this by trying to set the CONTEXT    *
*       to the NEWPLEX name. If successful (NEWPLEX already exists *
*       as a CONTEXT) issue messages and get out.                *
*-----*
EXEC  CPSM QUALIFY                                X
      CONTEXT(NEWPLEX)                          X
      THREAD(THREAD)                             X
      RESPONSE(RESPONSE)                         X
      REASON(REASON)
CLC  RESPONSE,EYUVALUE(OK)    RESPONSE OK?
BE   ERRNISPC                  Yes - already a CONTEXT

```

```

*-----*
*   Verify that the desired NEWPLEX name is not already a   *
*   CSYSDEF or CSYSGRP in the old, soon to be new, CICSplex. *
*   *
*   Here we will start issuing EXEC CPSM GET requests, to   *
*   get result sets of different Resource Tables. We make   *
*   the call through the GETOBJ subroutine. Variable OBJECT  *
*   must be set with the Resource Table name. If we only want *
*   a subset of the records for a given Resource Table, we also *
*   set variable CRITERIA with a selection criteria string.   *
*   This string can contain references to any fields in the   *
*   Resource Table, connected by logical operators, and must  *
*   end with a period - . -. Variable CRITLEN must be loaded *
*   with the length of the criteria string.                  *
*   *
*   We will check the RESPONSE from GET calls inline, instead *
*   of in the subroutine. The reason for this is that sometimes *
*   a RESPONSE of OK will mean that we have a problem (e.g., *
*   the NEWPLEX name already exists as a CICS System name).  *
*-----*
*
*   Ask for a CSYSSYS record equal to the NEWPLEX name.
*
MVC  OBJECT,=CL8'CSYSDEF'
MVC  CRITERIA(5),=CL5'NAME='
MVC  CRITERIA+5(8),NEWPLEX
MVI  CRITERIA+13,C'.'
MVC  CRITLEN,=F'14'
BAS  R10,GETOBJ          Build result set
CLC  RESPONSE,EYUVALUE(OK)  RESPONSE OK?
BE   ERRNISC             Yes - already a CICS system
CLC  RESPONSE,EYUVALUE(NODATA) No CSYSDEF with NEWPLEX name?
BE   NOTCSYS            Yes - continue
B    ERRGETO             No - some error - msgs and out
NOTCSYS DS  0H
*
*   Ask for a CSYSGRP record equal to the NEWPLEX name.
*
MVC  OBJECT,=CL8'CSYSGRP'
MVC  CRITERIA(6),=CL6'GROUP='
MVC  CRITERIA+6(8),NEWPLEX
MVI  CRITERIA+14,C'.'
MVC  CRITLEN,=F'15'
BAS  R10,GETOBJ          Build the result set
CLC  RESPONSE,EYUVALUE(OK)  RESPONSE OK?
BE   ERRNISS            Yes - already a system group
CLC  RESPONSE,EYUVALUE(NODATA) No CSYSGRP with NEWPLEX name?
BE   NOTCGRP            Yes - continue
B    ERRGETO             No - some error - msgs and out
NOTCGRP DS  0H
*-----*
*   If we have gotten this far, we know that NEWPLEX is not *
*   already the name of a CICSplex, CMAS, CICS System, or   *
*   CICS System group - so we can start building the NEWPLEX. *
*   *
*   Switch CONTEXT to MPCMAS to build NEWPLEX and add CMASs. *
*-----*
MVC  CONTEXT,MPCMAS

```

sample program EYUAAPI3

```

*-----*
*      Build new plex using OLDPLEX as a model.      *
*-----*
*      The record that defines a CICSplex is the CPLEXDEF Resource *
*      Table. We will GET the OLDPLEX CPLEXDEF record, modify *
*      it as needed, and then CREATE the NEWPLEX CPLEXDEF records. *
*      This creates the NEWPLEX. *
*-----*
MVI   PLEXBLT,C'N'          Indicate NEWPLEX not built yet
*
*      First GET CPLEXDEF record for the OLDPLEX.
*
MVC   OBJECT,=CL8'CPLEXDEF'
MVC   CRITERIA(9),=CL9'CICSplex='
MVC   CRITERIA+9(8),OLDPLEX
MVI   CRITERIA+17,C'.'
MVC   CRITLEN,=F'18'
BAS   R10,GETOBJ           Build result set
CLC   RESPONSE,EYUVALUE(OK) RESPONSE OK?
BNE   ERRGETO             No - msgs and out
*
*      Here we start using the GETBUF subroutine. This subroutine
*      GETMAINS a buffer into which we can FETCH the records of the
*      result set that we last issued a GET for.
*
BAS   R10,GETBUF           Get storage to receive recs
*
*      Here we start using the FETCH subroutine. This subroutine
*      reads all the records from the result set into the buffer.
*      On return to mainline, R2 points to the first record in
*      the buffer.
*
BAS   R10,FETCH           Sets R2 to fetched record
*
*      Change the OLDPLEX CPLEXDEF record into the NEWPLEX
*      CPLEXDEF record.
*
USING CPLEXDEF,R2         Map the record
MVC   CPLEXDEF_CICSplex,NEWPLEX          X
*                               Set CICSplex name to NEWPLEX
MVC   CPLEXDEF_DESC,=CL30'API cloned from' X
*                               Modify CICSplex ....
MVC   CPLEXDEF_DESC+16(8),OLDPLEX        X
*                               .... description
MVC   NEWPLXD(CPLEXDEF_TBL_LEN),0(R2)    X
*                               Save NEWPLEX def and len ....
MVC   NEWPLXDL,=A(CPLEXDEF_TBL_LEN)      X
*                               .... for possible later REMOVE

```

```

*
* Here we start using the CREATE subroutine. This subroutine
* will cause a CPSM Resource Table record to be built. Variable
* OBJECT needs to be preset to the Resource Table name, the
* Resource Table record to be built must be pointed to by R2
* and must be filled out prior to called CREATE.
*
  BAS R10,CREATE          CREATE NEWPLEX
  MVI PLEXBLT,C'Y'       Indicate NEWPLEX now built
*
* Here we start using the FREEBUF subroutine. This subroutine
* FREEMAINS the buffer into which we FETCHed the records.
*
  BAS R10,FREEBUF        Free record storage
*
* When a result set is built (in our program by either GET or
* PERFORM) an id is associated with the result set and placed
* into the variable pointed to by keyword RESULT (for GET we
* are using variable RESULT - for PERFORM, RESULT2). This is
* done so that subsequent calls can reference the result set
* built (e.g, FETCH can retrieve records for GET). When we
* are done using a result set, we must DISCARD it, so that
* CPSM frees us resources allocated for the result set.
* Note that we have not done this with the 2 previous GETs
* we did since the object of them was to NOT get a result set.
* If any of the previous GETs caused a result set to get built,
* we DISCONNECT from CPSM - which causes all our resources to
* be released - and exit.
*
  MVC RESULTD,RESULT     Copy GET result set id for      X
                        DISCARD
  BAS R10,DISCARD        Discard the GET result set
  DROP R2                Drop mapping to CPLEXDEF rec
*-----*
* Add CMASs in OLDPLEX to NEWPLEX.
*
* There is a CICSplex Resource Table record for each CMAS
* that participates in the management of a plex. We first
* ask for all the CICSplex records for OLDPLEX, and use
* this info to add the CMASs to the NEWPLEX.
*-----*
* Ask for the CICSplex records from the OLDPLEX.
*
  MVC OBJECT,=CL8'CICSplex'
  MVC CRITERIA(9),=CL9'PLEXNAME='
  MVC CRITERIA+9(8),OLDPLEX
  MVI CRITERIA+17,C'.'
  MVC CRITLEN,=F'18'
  BAS R10,GETOBJ         Build result set
  CLC RESPONSE,EYUVALUE(OK)  RESPONSE OK?
  BNE ERRGETO           no - msgs and out
  BAS R10,GETBUF        Get storage for records
  BAS R10,FETCH         Points R2 to first record
  USING CICSplex,R2     Map the Resource Table
  L R5,COUNT            Will loop the number of          X
                        returned CMASs
*
* The MP CMAS is added to the CICSplex when the CPLEXDEF
* record was CREATED. To add any other CMASs to the CICSplex
* we issue a PERFORM against the CPLEXDEF record for NEWPLEX,
* with a parm = CICSplex(newplex) CMAS(cmasname).
*
  MVC ADDCPARM(ADDCLEN),ADDC Build most of parm
  MVC PARMLEN,=A(ADDCLEN) Set its length
  MVC ADDCPLEX,NEWPLEX    Add CICSplex name to parm
  MVC OBJECT,=CL8'CPLEXDEF' PERFORM against CPLEXDEF
ADDCMAS DS 0H
  CLC CICSplex_CMASNAME,MPCMAS CMAS = MPCMAS?
  BE NOADDMP             Yes - don't add it then
  MVC ADDCCMAS,CICSplex_CMASNAME
                        Add CMAS name to PARM          X
                        This comes from the CICSplex    X
                        records.

```

sample program EYUAAPI3

```

*
*   Note that we already have the CICSPLEX result set active,
*   with the id in RESULT.  So here we will use RESULT2 for
*   result set that is built for each PERFORM.
*
MVC  RESULT2,=F'0'           Always build new result set
EXEC  CPSM PERFORM                               X
      OBJECT(OBJECT)                               X
      ACTION(=CL12'ASSIGN')                         X
      PARM(ADDCPARM)                                X
      PARMLN(PARMLN)                                X
      RESULT(RESULT2)                               X
      CONTEXT(CONTEXT)                             X
      THREAD(THREAD)                               X
      RESPONSE(RESPONSE)                           X
      REASON(REASON)
CLC  RESPONSE,EYUVALUE(OK)  RESPONSE OK?
BNE  ERRPERF              no - msgs and out
MVC  RESULTD,RESULT2      Copy PERFORM result set id for X
                          DISCARD
NOADDMP BAS  R10,DISCARD   Discard the PERFORM result set
DS      0H
*
*   We need to get to the next CICSPLEX record for the next CMAS.
*   The GETBUF subroutine places into variable RECLN the length
*   of the Resource Table record.  We now add this to the address
*   of the current record to point to the next record.
*
A     R2,RECLN
BCT  R5,ADDCMAS           Add the next CMAS
*
*   No more CICSPLEX records - discard the CICSPLEX result set
*   and continue on.
*
BAS  R10,FREEBUF         Free FETCHed record storage
MVC  OBJECT,=CL8'CICSPLEX' For possible DISCARD error msg
MVC  RESULTD,RESULT      Copy GET result set id for X
                          DISCARD
BAS  R10,DISCARD        Discard the GET result set
DROP R2                  Drop mapping to CICSPLEX rec

```

```

*-----*
*       Take all defs in OLDPLEX and put into NEWPLEX.           *
*                                                                 *
*       We have a list of all CICSplex Resource Table names. We  *
*       loop through this list, getting all the records for a    *
*       specific Resource Table from the OLDPLEX and adding them *
*       to the NEWPLEX.                                         *
*-----*
          MVC  CRITLEN,=F'0'           Want all records from each X
                                           Resource Table - so we don't X
                                           want a CRITERIA for GET.
          LA   R3,DEFNUM               Get number of Resource Tables
          LA   R4,DEFLIST              Point R4 to first Resource X
                                           Table in list
BLDLOOP DS   0H
          MVC  OBJECT,0(R4)            Move in Resource Table name
*
*       Get old data - set CONTEXT to OLDPLEX.
*
          MVC  CONTEXT,OLDPLEX
          MVC  SCOPE,OLDPLEX
          BAS  R10,GETOBJ               Build result set
          CLC  RESPONSE,EYUVALUE(OK)   RESPONSE OK?
          BE   GOTDEFS                  Yes - FETCH and add
          CLC  RESPONSE,EYUVALUE(NODATA) No records returned?
          BE   NODATA                    Yes - on to next Resource Tab
          B    ERRGETO                  GET error - msgs and out
          GOTDEFS DS   0H
          BAS  R10,GETBUF               Get storage for records
          BAS  R10,FETCH                 Point R2 to first record
          L    R5,COUNT                 Load number of records for loop
*
*       Add new data - set CONTEXT to NEWPLEX.
*
          MVC  CONTEXT,NEWPLEX
          CRELOOP DS   0H
*
*       We need to check if the object being created is a RTAINAPS
*       table. If it is, we need to check if the SCOPE is the
*       OLDPLEX name - and if so, change it to the NEWPLEX name.
*       The RTAINAPS table is the only resource table in our list
*       that may have the OLDPLEX specified as a SCOPE.
*
          CLC  OBJECT,=CL8'RTAINAPS'    Creating an RTAINAPS?
          BNE  CRELOOP2                 No, just CREATE it
          USING RTAINAPS,R2              May to the record
          CLC  RTAINAPS_SCOPE,OLDPLEX    Is SCOPE equal to OLDPLEX?
          BNE  CRELOOP2                 No, don't change record
          MVC  RTAINAPS_SCOPE,NEWPLEX    Alter SCOPE to NEWPLEX
          DROP R2                        Drop mapping to RTAINAPS rec
          CRELOOP2 DS   0H
          BAS  R10,CREATE                 CREATE record in NEWPLEX
          A    R2,RECLN                    Point to next record
          BCT  R5,CRELOOP                 Loop
          BAS  R10,FREEBUF                 Release record storage
          MVC  RESULTD,RESULT              Copy GET result set id for X
                                           DISCARD
          BAS  R10,DISCARD                 Discard the GET result set
          NODATA DS   0H
          LA   R4,8(,R4)                  Point to next Resource Table
          BCT  R3,BLDLOOP                 Do next Resource Table
*
*       We have gone through all the Resource Tables ok. Set
*       the return code to 0.
*
          MVC  RETCODE,=F'0'

```

sample program EYUAAPI3

```
*-----*
*      Disconnect the connection and exit the program.      *
*-----*
EXITDISC DS    0H
          EXEC  CPSM DISCONNECT
                                THREAD(THREAD)           X
                                RESPONSE(RESPONSE)        X
                                REASON(REASON)
EXIT      DS    0H
          CLOSE (SYSPRINT)
*-----*
*      Unchain save area, FREEMAIN working storage, and restore *
*      registers.                                             *
*-----*
          L     R2,RETCODE           Retrieve return code
          L     R13,4(,R13)
          L     R1,8(,R13)
          FREEMAIN R,A=(R1),LV=WORKLEN
          L     R14,12(,R13)
          LR    R15,R2
          LM    R0,R12,20(R13)
          LA   R15,0
          BR   R14
```



```

*-----*
*      Error routines.      *
*-----*
ERRCON  DS      0H
        MVC     OUTLINE,=CL80'Error: Connecting to the API'
        BAS     R9,PUTMSG
        BAS     R10,DORR          Format and msg RESPONSE/REASON
        B       EXIT             Exit
ERRNISPC DS      0H
        MVC     OUTLINE,=CL80'Error: NEWPLEX is already defined as a CICX
        SpIex or CMAS'
        BAS     R9,PUTMSG
        B       EXITDISC         DISCONNECT and exit
ERRNISC  DS      0H
        MVC     OUTLINE,=CL80'Error: NEWPLEX is already defined as a CICX
        S system in the OLDPLEX'
        BAS     R9,PUTMSG
        B       EXITDISC         DISCONNECT and exit
ERRNISS  DS      0H
        MVC     OUTLINE,=CL80'Error: NEWPLEX is already defined as a CICX
        S system group in the OLDPLEX'
        BAS     R9,PUTMSG
        B       EXITDISC         DISCONNECT and exit
ERRPERF  DS      0H
        MVC     OUTLINE,=CL80'Error: Adding a CMAS to the NEWPLEX'
        BAS     R9,PUTMSG
        MVC     OUTLINE,=CL80' '
        MVC     OUTTXT1,=CL10'CMASNAME:'
        MVC     OUTDAT1,ADDCCMAS
        BAS     R9,PUTMSG
        BAS     R10,DORR          Format and msg RESPONSE/REASON
        B       EXITERR
ERRGETO  DS      0H
        MVC     OUTLINE,=CL80'Error: GETting an object'
        BAS     R9,PUTMSG
        B       DOOBJMSG
ERRQUERY DS      0H
        MVC     OUTLINE,=CL80'Error: QUERYing a record size.'
        BAS     R9,PUTMSG
        B       DOOBJMSG
ERRFETCH DS      0H
        MVC     OUTLINE,=CL80'Error: FETCHing an object.'
        BAS     R9,PUTMSG
        B       DOOBJMSG
ERRCREAT DS      0H
        MVC     OUTLINE,=CL80'Error: CREATEing an object.'
        BAS     R9,PUTMSG
        B       DOOBJMSG
ERRDISCA DS      0H
        MVC     OUTLINE,=CL80'Error: DISCARDing object.'
        BAS     R9,PUTMSG
DOOBJMSG DS      0H
        MVC     OUTLINE,=CL80' '
        MVC     OUTTXT1,=CL10'OBJECT:'
        MVC     OUTDAT1,OBJECT
        BAS     R9,PUTMSG
        BAS     R10,DORR
EXITERR  DS      0H
        CLI     PLEXBLT,C'Y'      Did we CREATE the NEWPLEX?
        BNE     EXITDISC         No - just DISCONNECT and exit

```

sample program EYUAAPI3

```

*
*   We had already CREATED the NEWPLEX when an error occurred
*   so we want to delete the NEWPLEX before ending our program.
*
EXEC  CPSM REMOVE                                X
      OBJECT(=CL8'CPLEXDEF')                    X
      FROM(NEWPLXD)                             X
      LENGTH(NEWPLXDL)                          X
      CONTEXT(MPCMAS)                           X
      THREAD(THREAD)                            X
      RESPONSE(RESPONSE)                        X
      REASON(REASON)
CLC   RESPONSE,EYUVALUE(OK)   RESPONSE OK?
BE    EXITDISC               Yes - DISCONNECT and exit
MVC   OUTLINE,=CL80'Error: REMOVEing NEWPLEX.'
BAS   R9,PUTMSG
BAS   R10,DORR
B     EXITDISC                   DISCONNECT and exit
-----*
*   End of error routines.
*
-----*
*   Subroutines.
*
-----*
PUTMSG DS   0H
      PUT  SYSPRINT,OUTLINE
      BR   R9
DORR   DS   0H
-----*
*   Subroutine: DORR
*   Entry:      Via BAS R10,DORR
*   Function:   Put out error messages indicating what function
*              failed and the RESPONSE and REASON from that
*              function.
*   Processing: - Format the EXEC CPSM RESPONSE and move to the
*                OUTLINE.
*                - Format the EXEC CPSM REASON and move to the
*                OUTLINE.
*                - Call the PUTMSG subroutine to send the
*                RESPONSE/REASON data to SYSPRINT.
*                - Return to caller.
*
-----*
MVC   OUTLINE,=CL80' '      clear format area
MVC   OUTTXT1,=CL10'RESPONSE:' move in ....
L     R3,RESPONSE          load up the RESPONSE
CVD   R3,DOUBLE            convert to decimal
MVC   OUTDAT1(6),=XL6'402020202120' move in EDIT pattern
ED    OUTDAT1(6),DOUBLE+5  EDIT RESPONSE to format area
MVC   OUTTXT2,=CL10'REASON:' ... constant data
L     R3,REASON            load up the REASON
CVD   R3,DOUBLE            convert to decimal
MVC   OUTDAT2(6),=XL6'402020202120' move in EDIT pattern
ED    OUTDAT2(6),DOUBLE+5  EDIT REASON to format area
BAS   R9,PUTMSG            SEND it
MVC   OUTLINE,=CL80' '      clear out OUTLINE again
BAS   R9,PUTMSG            put out blank line
BR    R10                  return to caller

```

```

GETOBJ DS 0H
-----*
* Subroutine: GETOBJ *
* Entry: Via BAS R10,GETOBJ *
* Function: Issue the EXEC CPSM GET command to create a *
* result set for a specific object. Note that *
* all operands for GET must be preset in *
* mainline code - except for RESULT. *
* Processing: - Clear out the result set id - RESULT - so *
* that a new result set is always built. It *
* is the responsibility of mainline to DISCARD *
* any previous result set for GET. *
* - Determine if the GET request has a CRITERIA *
* and use the proper EXEC CPSM GET call. *
* - Note that GETOBJ does not check the RESPONSE *
* from CPSM - this is done in mainline. *
* - Return to caller. *
-----*
MVC RESULT,=F'0' Always get new result set
CLC CRITLEN,=F'0'
BE GETNOCRT
EXEC CPSM GET X
OBJECT(OBJECT) X
CRITERIA(CRITERIA) X
LENGTH(CRITLEN) X
COUNT(COUNT) X
RESULT(RESULT) X
THREAD(THREAD) X
CONTEXT(CONTEXT) X
RESPONSE(RESPONSE) X
REASON(REASON) X
BR R10
GETNOCRT DS 0H
EXEC CPSM GET X
OBJECT(OBJECT) X
COUNT(COUNT) X
RESULT(RESULT) X
THREAD(THREAD) X
CONTEXT(CONTEXT) X
RESPONSE(RESPONSE) X
REASON(REASON) X
BR R10

```

sample program EYUAAPI3

```

GETBUF  DS  0H
*-----*
*      Subroutine: GETBUF                                *
*      Entry:      Via BAS R10,GETBUF                    *
*      Function:   Get a buffer to hold all the records contained *
*                  in the last result set we build though GET. *
*      Processing: - Issue EXEC CPSM QUERY to get the length of *
*                  the Resource Table record. We use the same *
*                  OBJECT and RESULT from the GET. Variable *
*                  RECLLEN gets the record length. *
*                  - Check the RESPONSE from QUERY and issue msgs *
*                  and EXIT if not OK. *
*                  - Multiple the RECLLEN times the COUNT (returned *
*                  from last GET) to determine the buffer size *
*                  required and GETMAIN it. *
*                  - Save the buffer length (BUFLEN) and buffer *
*                  address (BUFFER) for the FREEMAIN call in *
*                  the FREEBUF subroutine. *
*                  - Return to caller. *
*-----*
EXEC  CPSM QUERY                                X
      OBJECT(OBJECT)                            X
      DATALENGTH(RECLLEN)                       X
      RESULT(RESULT)                             X
      THREAD(THREAD)                             X
      RESPONSE(RESPONSE)                         X
      REASON(REASON)
CLC  RESPONSE,EYUVALUE(OK)  RESPONSE OK?
BNE  ERRQUERY              No - msgs and out
L    R0,RECLLEN
L    R1,COUNT
MR   R0,R0
GETMAIN R,LV=(R1)
ST   R0,BUFLEN
ST   R1,BUFFER
BR   R10
FREEBUF DS  0H
*-----*
*      Subroutine: FREEBUF                                *
*      Entry:      Via BAS R10,FREEBUF                    *
*      Function:   To FREEMAIN the buffer created to hold the *
*                  records from the last result set we built . *
*                  through GET. *
*      Processing: - Use BUFLEN and BUFFER from GETBUF, FREEMAIN *
*                  the buffer area. *
*                  - Return to caller. *
*-----*
L    R0,BUFLEN
L    R1,BUFFER
FREEMAIN R,A=(R1),LV=(R0)
BR   R10

```


sample program EYUAAPI3

```

CREATE  DS  0H
*-----*
*      Subroutine: CREATE                                *
*      Entry:      Via BAS R10,CREATE                    *
*      Function:   Issue the EXEC CPSM CREATE to build a Resource *
*                  Table record.                        *
*      Processing: - Place the length of the record to be build *
*                  (RECLen from GETBUF) into variable LENGTH. *
*                  R2 should have been set by mainline to point *
*                  to the record itself.                *
*                  - When CREATEing a LNKxxCG record (spec to *
*                  group link) we need to specify a parm - *
*                  NONE. - to indicate that we only want the *
*                  CREATE to associate the spec to the group. *
*                  Any systems in the group that need to be *
*                  added to the spec have already been done *
*                  by CREATE of LNKxxCS records (spec to *
*                  system link). If this is a LNKxxCG record, *
*                  set the PARM and PARMLenGth.         *
*                  - Issue the proper format of EXEC CPSM CREATE *
*                  (either with PARM/PARMLen or without). *
*                  - Check the RESPONSE - if not OK, issue msgs *
*                  and exit.                             *
*                  - Return to caller.                  *
*-----*
MVC  LENGTH,RECLen
CLC  OBJECT(4),=CL4'LNKS'
BNE  CRENOPRM
CLC  OBJECT+6(2),=CL2'CG'
BNE  CRENOPRM
MVC  PARM,=CL5'NONE.'
MVC  PARMLen,=F'5'
EXEC  CPSM CREATE                                X
      OBJECT(OBJECT)                            X
      FROM(0(,R2))                              X
      LENGTH(LENGTH)                            X
      PARM(PARM)                                X
      PARMLen(PARMLen)                          X
      THREAD(THREAD)                            X
      CONTEXT(CONTEXT)                          X
      RESPONSE(RESPONSE)                        X
      REASON(REASON)
      B      CRECHKRR
CRENOPRM DS  0H
EXEC  CPSM CREATE                                X
      OBJECT(OBJECT)                            X
      FROM(0(,R2))                              X
      LENGTH(LENGTH)                            X
      THREAD(THREAD)                            X
      CONTEXT(CONTEXT)                          X
      RESPONSE(RESPONSE)                        X
      REASON(REASON)
CRECHKRR DS  0H
CLC  RESPONSE,EYUVALUE(OK)
BNE  ERRCREAT
BR   R10

```

```

DISCARD DS 0H
*-----*
* Subroutine: DISCARD *
* Entry: Via BAS R10,DISCARD *
* Function: Issue the EXEC CPSM DISCARD to discard a result *
* set built by CPSM. In our program, both GET *
* and PERFORM build result sets. *
* Processing: - Issue EXEC CPSM DISCARD for the result set. *
* The result set id must be placed into *
* RESULTD by mainline. *
* - Check the RESPONSE - if not OK, issue msgs *
* and exit. *
* - Return to caller. *
*-----*
EXEC CPSM DISCARD X
RESULT(RESLTD) X
THREAD(THREAD) X
RESPONSE(RESPONSE) X
REASON(REASON)
CLC RESPONSE,EYUVALUE(OK)
BNE ERRDISCA
BR R10
*-----*
* End of subroutines. *
*-----*

```

sample program EYUAAPI3

```

*-----*
*   Following is a list of all CPSM Resource Tables that can   *
*   be part of a CICSplex.  The order that they are in (which *
*   is the order they will be built in our program) is        *
*   important, since some Resource Tables will reference other *
*   Resource Tables previously built.  The following list is   *
*   complete and the order OK for the current release of      *
*   CPSM (V1R3M0).                                           *
*-----*
DEFLIST DS    0C
        DC    CL8'PERIODEF'   Time period definitions
        DC    CL8'ACTION  '   RTA action definitions
        DC    CL8'CSYSDEF  '   CICS system definitions
        DC    CL8'CSYSGRP '   CICS system group definitions
        DC    CL8'CSGLCGCS'   CICS systems in groups links
        DC    CL8'CSGLCGCG'   CICS groups in groups links
        DC    CL8'MONDEF   '   Monitor definitions
        DC    CL8'MONGROUP'   MON group definitions
        DC    CL8'MONSPEC  '   MON specification definitions
        DC    CL8'MONINGRP'   MON def in MON group links
        DC    CL8'MONINSPC'   MON spec to MON group links
        DC    CL8'LNKSMSCS'   MON spec to CICS system links
        DC    CL8'LNKSMSCG'   MON spec to CICS group links
        DC    CL8'EVALDEF  '   RTA evaluation definitions
        DC    CL8'RTADEF   '   Real time analysis definitions
        DC    CL8'STATDEF  '   User status probe definitions
        DC    CL8'RTAGROUP'   RTA group definitions
        DC    CL8'RTASPEC  '   RTA specification definitions
        DC    CL8'RTAINGRP'   RTADEF in RTA group links
        DC    CL8'STAINGRP'   STATDEF in RTA group links
        DC    CL8'RTAINSPC'   RTA spec to RTA group links
        DC    CL8'LNKSRSCS'   RTA spec to CICS group links
        DC    CL8'LNKSRSCG'   RTA spec to CICS system links
        DC    CL8'APSPEC   '   RTA/APM specification defs
        DC    CL8'RTAINAPS'   RTA/APM spec to RTA group links
        DC    CL8'CMDMPAPS'   RTA spec to primary CMAS links
        DC    CL8'CMDMSAPS'   RTA spec to secondary CMAS links
        DC    CL8'TRANGRP  '   transaction group definitions
        DC    CL8'WLMDEF   '   Workload definitions
        DC    CL8'WLMGROUP'   WLM group definitions
        DC    CL8'WLMSPEC  '   WLM specification definitions
        DC    CL8'DTRINGRP'   Transactions in trangrp links
        DC    CL8'WLMINGRP'   WLM def in WLM group links
        DC    CL8'WLMINSPC'   WLM spec to WLM group links
        DC    CL8'LNKSWSCS'   WLM spec to CICS group links
        DC    CL8'LNKSWSCG'   WLM spec to CICS system links
DEFNUM  EQU    (*-DEFLIST)/8
ADDC    DS    0X
        DC    CL09'CICSplex('
        DC    CL08'  '
        DC    CL07') CMAS('
        DC    CL08'  '
        DC    CL02').'
ADDCLN  EQU    *-ADDC
SYSPRINT DCB   DDNAME=SYSPRINT,DSORG=PS,MACRF=PM
WORKSTOR DSECT
SAVEAREA DS    18F
DFHEIPL  DS    13F
         DS    51F

```



```

DOUBLE DS D
RETCODE DS F
RESPONSE DS F
REASON DS F
THREAD DS F
RESULT DS F
RESULT2 DS F
RESULTD DS F
COUNT DS F
LENGTH DS F
PARMLN DS F
BUFLN DS F
BUFFER DS F
RECLN DS F
NEWPLXDL DS F
CRITLEN DS F
CRITERIA DS CL80
CONTEXT DS CL8
SCOPE DS CL8
OBJECT DS CL8
OLDPLEX DS CL8
NEWPLEX DS CL8
MPCMAS DS CL8
OUTLINE DS 0CL80
OUTTXT1 DS CL10
OUTDAT1 DS CL8
          DS CL2
OUTTXT2 DS CL10
OUTDAT2 DS CL8
          DS CL42
PARM DS CL5
PLEXBLT DS CL1
ADDCPARM DS 0XL(ADDCLN)
          DS CL09
ADDCPLEX DS CL08
          DS CL07
ADDCCMAS DS CL08
          DS CL02
          DS D
NEWPLXD DS XL(CPLEXDEF_TBL_LEN)
WORKLEN EQU *-WORKSTOR
          COPY CPLEXDEF
          COPY CICSplex
          COPY RTAINAPS
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
END EYUAAPI3

```

Sample program EYULAPI4

Program EYULAPI4 is written in COBOL for the CICS environment.

EYUxAPI4 This program does the following:

- Establishes a connect to the API.
- Creates a BAS definition for a TS Model (TSMDEF) specifying a version of 1.
- Creates a result set containing the previously defined TSMDEF.
- Issues a PERFORM OBJECT command to INSTALL the TSMDEF into the target scope.
- Terminates the API connection.
- BAS errors are processed using BINCONRS, BINCONSC, and BINSTERR resource table records.

Commands Used: CONNECT, CREATE, GET, PERFORM, OBJECT, TERMINATE, FEEDBACK, FETCH TERMINATE, TRANSLATE

```

IDENTIFICATION DIVISION.
PROGRAM-ID. EYULAPI4
*****
*
* MODULE NAME = EYULAPI4
*
* DESCRIPTIVE NAME = CPSM SAMPLE API PROGRAM 4
*                   (SAMPLE COBOL VERSION)
*
* COPYRIGHT = Licensed Materials - Property of IBM
*            5695-081
*            (C) Copyright IBM Corp. 1995, 1997
*            All Rights Reserved
*
*            US Government Users Restricted Rights - Use,
*            duplication or disclosure restricted by GSA ADP
*            Schedule Contract with IBM Corp.
*
* STATUS = %CP00
*
* FUNCTION =
*
* TO PROVIDE AN EXAMPLE OF THE USE OF THE FOLLOWING EXEC CPSM
* COMMANDS: CONNECT, CREATE, FEEDBACK, FETCH, GET,
*          PERFORM OBJECT, TERMINATE.
*
* WHEN INVOKED, THE PROGRAM DEPENDS UPON THE VALUES HELD IN THE
* W-CONTEXT AND W-SCOPE DECLARATIONS WHEN ESTABLISHING A
* CONNECTION WITH CICSplex SM. THEY MUST TAKE THE FOLLOWING
* VALUES:
*
* W-CONTEXT = THE NAME OF A CMAS OR CICSplex. REFER TO THE
*            DESCRIPTION OF THE EXEC CPSM CONNECT COMMAND
*            FOR FURTHER INFORMATION REGARDING THE CONTEXT
*            OPTION.
*
* W-SCOPE   = THE NAME OF A CICSplex, CICS SYSTEM, OR CICS
*            SYSTEM GROUP WITHIN THE CICSplex. REFER TO THE
*            DESCRIPTION OF THE EXEC CPSM CONNECT COMMAND
*            FOR FURTHER INFORMATION REGARDING THE SCOPE
*            OPTION.
*
* THIS SAMPLE REQUIRES NO PARAMETERS AT INVOCATION TIME.
*
* WHEN CREATING THE BAS DEFINITION THE PROGRAM DEPENDS UPON THE
* VALUES HELD IN THE W-DEFNAME AND W-DEFPREFIX DECLARATIONS.
* THEY MUST TAKE THE FOLLOWING VALUES:
*
* W-DEFNAME = THE NAME OF THE CREATED BAS DEFINITION. A
*            1 TO 8 CHARACTER VALUE.
*
* W-DEFPFIX = THE MODEL PREFIX OF THE CREATED BAS DEFINITION.
*            A 1 TO 16 CHARACTER VALUE.
*
*
*

```

sample program EYULAPI4

```
* WHEN INSTALLING THE BAS DEFINITION THE PROGRAM USES THE      *
* VALUE HELD IN THE W-TSCOPE DECLARATION AS THE TARGET FOR    *
* THE INSTALL OPERATION. IT MUST TAKE THE FOLLOWING VALUE :   *
*                                                              *
* W-TSCOPE   = THE NAME OF A CICS SYSTEM, OR CICS             *
*              SYSTEM GROUP WITHIN THE CICSplex. REFER TO THE  *
*              DESCRIPTION OF THE TARGET PARAMETER OF AN      *
*              INSTALL ACTION IN THE RESOURCE TABLE REFERENCE *
*              FOR FURTHER INFORMATION REGARDING THE TARGET    *
*              SCOPE VALUE.                                   *
*                                                              *
* THE SAMPLE ESTABLISHES AN API CONNECTION AND ISSUES A CREATE *
* COMMAND TO CREATE A BAS DEFINITION. A GET COMMAND IS ISSUED *
* TO OBTAIN A RESULT SET CONTAINING THE CREATED BAS DEFINITION.*
*                                                              *
* USING THE PERFORM OBJECT ACTION(INSTALL) COMMAND EACH RECORD *
* IN THE RESULT SET IS INSTALLED INTO THE TARGET SCOPE        *
* IDENTIFIED BY THE W-TSCOPE DECLARATION.                     *
*                                                              *
* FINALLY, THE API CONNECTION IS TERMINATED.                   *
*                                                              *
* ANY BAS ERRORS ARE REPORTED USING THE BINCONRS, BINCONSC, AND *
* BINSTERR RESOURCE TABLES.                                   *
*                                                              *
```

```
* NOTES :                                                       *
* DEPENDENCIES = S/390, CICS                                   *
* RESTRICTIONS = NONE                                         *
* REGISTER CONVENTIONS =                                       *
* MODULE TYPE   = EXECUTABLE                                   *
* PROCESSOR     = COBOL                                        *
* ATTRIBUTES    = READ ONLY, SERIALY REUSABLE                 *
*                                                         *
* -----*
* ENTRY POINT = EYULAPI4                                       *
*                                                         *
* PURPOSE = ALL FUNCTIONS.                                       *
*                                                         *
* LINKAGE = FROM CICS EITHER WITH EXEC CICS LINK OR AS A CICS  *
*           TRANSACTION.                                         *
*                                                         *
* INPUT   = NONE.                                               *
*                                                         *
* -----*
* ENVIRONMENT DIVISION.
* DATA DIVISION.
* WORKING-STORAGE SECTION.
* -----*
*   CHANGE W-CONTEXT AND W-TSCOPE TO MATCH YOUR INSTALLATION *
*   CHANGE W-DEFNAME AND W-DEFPFIX FOR THE CREATE COMMAND.   *
*   CHANGE W-TSCOPE FOR THE PERFORM OBJECT COMMAND.         *
* -----*
01 W-CONTEXT      PIC X(8) VALUE 'RTGA  ' .
01 W-TSCOPE       PIC X(8) VALUE 'RTGA  ' .
01 W-DEFNAME      PIC X(8) VALUE 'EYULAPI4' .
01 W-DEFPFIX     PIC X(16) VALUE 'EYUL*          ' .
01 W-TSCOPE      PIC X(8) VALUE 'RTGF  ' .
* -----*
```

```

01 W-RESPONSE      PIC S9(8) USAGE BINARY.
01 W-REASON        PIC S9(8) USAGE BINARY.
01 W-BUFFER        PIC X(32767).
01 W-BUFFERLEN     PIC S9(8) COMP.
01 W-FBUBUFF       PIC X(248).
01 W-FBTTKN        PIC S9(8) COMP.
01 W-THREAD        PIC S9(8) USAGE BINARY.
01 W-RESULT        PIC S9(8) USAGE BINARY.
01 W-RECCNT        PIC S9(8) USAGE BINARY.
01 W-CRITERIA      PIC X(80) VALUE SPACES.
01 W-CRITERIALEN   PIC S9(8) USAGE BINARY.
01 W-PARM          PIC X(80) VALUE SPACES.
01 W-PARMLN        PIC S9(8) USAGE BINARY.
01 W-MSG-TEXT.
  02 W-TEXT         PIC X(80) VALUE SPACES.
  02 W-LINECTL      PIC X(1) VALUE X'13'.
01 ARRAYS.
  02 CH8ARR         OCCURS 20 TIMES PIC X(8).
  02 FULLARR        OCCURS 60 TIMES PIC S9(8) COMP.
01 III             PIC S9(8) VALUE ZERO.
01 CODEV           PIC S9(8) COMP.
01 CHARV           PIC X(12).
01 LASTCMD        PIC X(20).
01 LASTTHR        PIC S9(8) COMP.
01 LASTRES        PIC S9(8) COMP VALUE 0.
01 BINZERO        PIC X(1) VALUE X'00'.
01 BLNKPAD        PIC X(40)
  VALUE '
01 FBCHAR2        PIC X(2).
01 FBHALF4        REDEFINES FBCHAR2.
  02 FBHALF        PIC S9(4) COMP.
01 PICZZZ9A       PIC ZZZ9.
01 PICZZZ9B       PIC ZZZ9.
01 PICZZZ9        PIC ZZZ9.
01 PYCZZZ9        PIC ZZZ9.
01 PIKZZZ9        PIC ZZZ9.
01 PYKZZZ9        PIC ZZZ9.
01 PICZZZZZZ9     PIC ZZZZZZZ9.
01 CHR8           PIC X(8).
01 CHR12          PIC X(12).
01 CHAR6          PIC X(6).
01 CHAR12         PIC X(12).
* Include the resource table copybooks...
COPY TSMDEF.
COPY FEEDBACK.
COPY BINCONRS.
COPY BINCONSC.
COPY BINSTERR.

```

sample program EYULAPI4

```
*****
* Start of LINKAGE section *
*****
LINKAGE SECTION.

PROCEDURE DIVISION.
EYULAPI4-START SECTION.
EYULAPI4-00.

*-----*
*   OBTAIN A CPSM API CONNECTION.                               *
*                                                                 *
*   THE API WILL RETURN A TOKEN IDENTIFYING THE THREAD IN      *
*   VARIABLE W-THREAD.                                         *
*-----*
MOVE 'Establishing Connection...' TO W-TEXT.
* DISPLAY W-TEXT.
EXEC CICS SEND FROM(W-TEXT) LENGTH(81) ERASE END-EXEC.
EXEC CPSM CONNECT
      CONTEXT(W-CONTEXT)
      SCOPE(W-SCOPE)
      VERSION('0140')
      THREAD(W-THREAD)
      RESPONSE(W-RESPONSE)
      REASON(W-REASON)
END-EXEC.
IF W-RESPONSE NOT = EYUVALUE(OK) GO TO NO-CONNECT.

*-----*
*   CREATE A TS MODEL DEFINITION (TSMDEF)                       *
*                                                                 *
*   A TSMDEF is created with a version of 1.                   *
*-----*
INITIALIZE TSMDEF.
MOVE X'01' TO DEFVER OF TSMDEF.
MOVE W-DEFNAME TO NAME-R OF TSMDEF.
MOVE W-DEFPFX TO PREFIX OF TSMDEF.
MOVE DFHVALUE(AUXILIARY) TO LOCATION OF TSMDEF.
MOVE EYUVALUE(NO) TO RECOVERY OF TSMDEF.
MOVE EYUVALUE(NO) TO SECURITY-R OF TSMDEF.
MOVE 'Sample TSMDEF definition' TO DESCRIPTION OF TSMDEF.
* Copy the definition into our buffer...
MOVE TSMDEF TO W-BUFFER.
MOVE TSMDEF-TBL-LEN TO W-BUFFERLEN.
MOVE 'Creating TSMDEF...' TO W-TEXT.
* DISPLAY W-TEXT.
EXEC CICS SEND FROM(W-TEXT) LENGTH(81) WAIT END-EXEC.
EXEC CPSM CREATE
      OBJECT('TSMDEF')
      FROM(W-BUFFER)
      LENGTH(W-BUFFERLEN)
      THREAD(W-THREAD)
      RESPONSE(W-RESPONSE)
      REASON(W-REASON)
END-EXEC.
MOVE 'CREATE' TO LASTCMD.
MOVE W-THREAD TO LASTTHR.
MOVE 0 TO LASTRES.
IF W-RESPONSE NOT = EYUVALUE(OK) GO TO UNEXPECTED.
```

```

*-----*
*   GET THE TSMDEF RESOURCE TABLE.                               *
*                                                                 *
*   CREATE A RESULT SET CONTAINING ENTRIES FOR ALL TSMDEFS      *
*   WITH NAMES EQUAL TO THE VALUE OF W-DEFNAME. .             *
*   THE NUMBER OF ENTRIES MEETING THE CRITERIA IS RETURNED    *
*   IN VARIABLE W-RECCNT.                                       *
*-----*
MOVE 'Get the created TSMDEF Resource Table...' TO W-TEXT.
* DISPLAY W-TEXT.
EXEC CICS SEND FROM(W-TEXT) LENGTH(81) WAIT END-EXEC.
STRING 'NAME=' DELIMITED BY SIZE
       W-DEFNAME DELIMITED BY SIZE
       '.' DELIMITED BY SIZE
       INTO W-CRITERIA.
MOVE LENGTH OF W-CRITERIA TO W-CRITERIALEN.
MOVE BINZERO TO W-RESULT.
EXEC CPSM GET OBJECT('TSMDEF')
          CRITERIA(W-CRITERIA)
          LENGTH(W-CRITERIALEN)
          COUNT(W-RECCNT)
          RESULT(W-RESULT)
          THREAD(W-THREAD)
          RESPONSE(W-RESPONSE)
          REASON(W-REASON)

END-EXEC.
IF W-RESPONSE NOT = EYUVALUE(OK) GO TO NO-GET.

*-----*
*   INSTALL EACH RECORD INTO THE SCOPE IDENTIFIED BY THE      *
*   VALUE OF W-TSCOPE.                                         *
*-----*
MOVE W-RECCNT TO PICZZZZZZ9.
STRING 'Installing ' DELIMITED BY SIZE
       PICZZZZZZ9 DELIMITED BY SIZE
       ' TSMDEF Entries...' DELIMITED BY SIZE
       INTO W-TEXT.
* DISPLAY W-TEXT
EXEC CICS SEND FROM(W-TEXT) LENGTH(81) WAIT END-EXEC.
STRING '(USAGE(LOCAL) TARGET(' DELIMITED BY SIZE
       W-TSCOPE DELIMITED BY SIZE
       ')).' DELIMITED BY SIZE
       INTO W-PARM.
MOVE LENGTH OF W-PARM TO W-PARMLN.

EXEC CPSM PERFORM OBJECT('TSMDEF')
          ACTION('INSTALL')
          PARM(W-PARM)
          PARMLN(W-PARMLN)
          RESULT(W-RESULT)
          THREAD(W-THREAD)
          RESPONSE(W-RESPONSE)
          REASON(W-REASON)

END-EXEC.
MOVE 'PERFORM OBJECT' TO LASTCMD.
MOVE W-THREAD TO LASTTHR.
MOVE W-RESULT TO LASTRES.
IF W-RESPONSE NOT = EYUVALUE(OK) GO TO UNEXPECTED.

MOVE 'Completed. Remove TSMDEF to re-run.' TO W-TEXT.
GO TO SCRNLG2.

```

```

*****
* Branch here if an unexpected CPSM error occurs *
*****
UNEXPECTED.
    MOVE W-RESPONSE TO PICZZZ9.
    STRING '*** RESPONSE=' DELIMITED BY SIZE PICZZZ9
    DELIMITED BY SIZE BLNKPAD DELIMITED BY SIZE INTO W-TEXT.
    PERFORM SCRNLG2.
    MOVE W-REASON TO PICZZZ9.
    STRING '*** REASON=' DELIMITED BY SIZE PICZZZ9
    DELIMITED BY SIZE BLNKPAD DELIMITED BY SIZE INTO W-TEXT.
    PERFORM SCRNLG2.
    MOVE '*** Unexpected error condition arose' TO W-TEXT.
    PERFORM SCRNLG2.
* Obtain FEEDBACK information
    IF LASTCMD = 'DISCONNECT' GO TO NOFEED.
    IF LASTCMD = 'FEEDBACK' GO TO NOFEED.
    IF LASTCMD = 'TERMINATE' GO TO NOFEED.
    STRING
    '*** Getting FEEDBACK data for ' DELIMITED BY SIZE
    LASTCMD DELIMITED BY SIZE
    INTO W-TEXT.
    PERFORM SCRNLG2.
    STRING
    BLNKPAD DELIMITED BY SIZE
    BLNKPAD DELIMITED BY SIZE
    INTO W-TEXT.
* Get the FEEDBACK data
    GETFEED.
* Clear error result set count
    MOVE 0 TO FULLARR(1).
    PERFORM GETFB THROUGH EGETFB
* Display FEEDBACK information
* Display information
    IF W-RESPONSE = EYUVALUE(OK)
        PERFORM DISPFEED
        IF FULLARR(1) NOT = 0 PERFORM GETFERT THROUGH EGETFER END-I
-F
        IF LASTRES NOT = 0 GO TO GETFEED END-IF
        MOVE '*** End of FEEDBACK data' TO W-TEXT
        PERFORM SCRNLG2
        GO TO NOFEED
    END-IF.
    MOVE W-RESPONSE TO PICZZZ9.
    MOVE W-REASON TO PYCZZZ9.
    STRING '*** FEEDBACK not available (' DELIMITED BY SIZE
    PICZZZ9 DELIMITED BY SIZE ',' DELIMITED BY SIZE
    PYCZZZ9 DELIMITED BY SIZE ')' DELIMITED BY SIZE
    BLNKPAD DELIMITED BY SIZE INTO W-TEXT END-STRING.
    PERFORM SCRNLG2.
NOFEED.
    EXEC CICS DELAY FOR SECONDS(10) END-EXEC.
* Exit from test case
    EXEC CICS RETURN END-EXEC.
    GOBACK.
    EXIT.

```



```

*****
* This subroutine obtains the FEEDBACK data *
*****
GETFB.
* Use exact buffer size
  MOVE FEEDBACK-TBL-LEN TO W-BUFFERLEN.
  IF LASTRES = 0 GO TO NORESULT.
RESULT.
  EXEC CPSM FEEDBACK
      INTO(W-FBBUFF) LENGTH(W-BUFFERLEN)
      RESULT(LASTRES)
      THREAD(LASTTHR)
      RESPONSE(W-RESPONSE)
      REASON(W-REASON)
  END-EXEC.

* If command didn't execute, get FEEDBACK no result set
* Command didn't execute?
  IF W-RESPONSE = EYUVALUE(NODATA)
    MOVE 0 TO LASTRES
    GO TO NORESULT
  END-IF.
  GO TO ENDFBACK.
NORESULT.
* Use exact buffer size
  MOVE FEEDBACK-TBL-LEN TO W-BUFFERLEN.
  EXEC CPSM FEEDBACK
      INTO(W-FBBUFF) LENGTH(W-BUFFERLEN)
      THREAD(LASTTHR)
      RESPONSE(W-RESPONSE)
      REASON(W-REASON)
  END-EXEC.

ENDFBACK.
EGETFB.
EXIT.

```

```

*****
* Branch here if FEEDBACK Error Result Token available *
*****
GETFERT.
  MOVE ERR-OBJECT OF FEEDBACK TO CH8ARR(1).
  STRING
  '*** Getting ' DELIMITED BY SIZE
  CH8ARR(1) DELIMITED BY SIZE
  ' error result set data for FEEDBACK' DELIMITED BY SIZE
  INTO W-TEXT.
  PERFORM SCRNLG2.
FERTRES.
* Use largest buffer size
  MOVE FEEDBACK-TBL-LEN TO W-BUFFERLEN.
  EXEC CPSM FETCH
        INTO(W-BUFFER) LENGTH(W-BUFFERLEN)
        RESULT(ERR-RESULT OF FEEDBACK)
        THREAD(LASTTHR)
        RESPONSE(W-RESPONSE)
        REASON(W-REASON)
  END-EXEC.

* Display FEEDBACK Error Result Token information
* Display information
  IF W-RESPONSE = EYUVALUE(OK)
    IF CH8ARR(1)= 'FEEDBACK'
      MOVE W-BUFFER TO W-FBBUFF
      PERFORM DISPFEED
    END-IF
    IF CH8ARR(1)= 'BINSTERR'
      PERFORM DISPBIER
    END-IF
    IF CH8ARR(1)= 'BINCONRS'
      PERFORM DISPBIRS
    END-IF
    IF CH8ARR(1)= 'BINCONSC'
      PERFORM DISPBISC
    END-IF
    GO TO FERTRES
  END-IF.
  MOVE W-RESPONSE TO PICZZZ9.
  MOVE W-REASON TO PYCZZZ9.
  STRING '*** FEEDBACK not available (' DELIMITED BY SIZE
  PICZZZ9 DELIMITED BY SIZE ',' DELIMITED BY SIZE
  PYCZZZ9 DELIMITED BY SIZE ')' DELIMITED BY SIZE
  BLNKPAD DELIMITED BY SIZE INTO W-TEXT END-STRING.
  PERFORM SCRNLG2.
EGETFER.
EXIT.

```

```
*****
* This subroutine displays FEEDBACK information *
*****
DISPFEEED.
```

```
MOVE W-FBBUFF TO FEEDBACK.
STRING BINZERO COMMAND OF FEEDBACK DELIMITED BY SIZE
INTO FBCHAR2.
MOVE FBHALF TO PICZZZ9.
MOVE RESPONSE OF FEEDBACK TO PYCZZZ9.
MOVE REASON OF FEEDBACK TO PIKZZZ9.
MOVE RSLTRECID OF FEEDBACK TO PYKZZZ9.
MOVE SPACES TO W-TEXT.
STRING 'Cmd=' PICZZZ9 ' Attr=' ATTRDATAVAL OF
FEEDBACK ' Eib=' CEIBDATAVAL OF FEEDBACK ' Err='
ERRCODEVAL OF FEEDBACK ' Rspn=' PYCZZZ9 ' Reas='
PIKZZZ9 ' ResId=' PYKZZZ9
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE ERROR-CODE OF FEEDBACK TO PICZZZ9.
MOVE CEIBRESP OF FEEDBACK TO PYCZZZ9.
MOVE CEIBRESP1 OF FEEDBACK TO PIKZZZ9.
MOVE CEIBFN OF FEEDBACK TO PYKZZZ9.
MOVE SPACES TO W-TEXT.
STRING ' ECode=' PICZZZ9 ' RESP=' PYCZZZ9
' RESP1=' PIKZZZ9 ' EibFn=' PYKZZZ9 ' Obj='
OBJECT-A OF FEEDBACK ' OAct=' OBJECT-ACT OF FEEDBACK
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE SPACES TO W-TEXT.
STRING ' Att1=' ATTR-NM1 OF FEEDBACK ' 2='
ATTR-NM2 OF FEEDBACK ' 3=' ATTR-NM3 OF FEEDBACK
' 4=' ATTR-NM4 OF FEEDBACK ' 5=' ATTR-NM5 OF
FEEDBACK DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE ERR-COUNT OF FEEDBACK TO PICZZZ9.
MOVE SPACES TO W-TEXT.
STRING ' FObj=' ERR-OBJECT OF FEEDBACK
' FCnt=' PICZZZ9
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE ERR-COUNT OF FEEDBACK TO FULLARR(1).
EXIT.
```

```
*****
* This subroutine displays BINSTERR information *
*****
DISPBIER.
```

```
MOVE W-BUFFER TO BINSTERR.
MOVE SPACES TO W-TEXT.
STRING 'CMAS=' CMASNAME OF BINSTERR ' Plex='
PLEXNAME OF BINSTERR ' CSys=' CICSNAME OF BINSTERR
' ResName=' RESNAME OF BINSTERR
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE RESVER OF BINSTERR TO PICZZZ9.
MOVE ERRCODE OF BINSTERR TO PYCZZZ9.
MOVE CRESP1 OF BINSTERR TO PIKZZZ9.
MOVE CRESP2 OF BINSTERR TO PYKZZZ9.
MOVE SPACES TO W-TEXT.
STRING ' ResVer=' PICZZZ9 ' ECode=' PYCZZZ9
' RESP=' PIKZZZ9 ' RESP1=' PYKZZZ9
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE CEIBFN OF BINSTERR TO PICZZZ9.
MOVE SPACES TO W-TEXT.
STRING ' EibFn=' PICZZZ9
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
EXIT.
```

```

*****
* This subroutine displays BINCONRS information *
*****
DISPBIRS.
    MOVE W-BUFFER TO BINCONRS.
    MOVE ERROP OF BINCONRS TO PICZZZ9.
    MOVE SPACES TO W-TEXT.
    STRING 'CMAS=' CMASNAME OF BINCONRS ' Plex='
    PLEXNAME OF BINCONRS ' Csys=' CICSNAME OF BINCONRS
    ' ResType=' RESTYPE OF BINCONRS ' EOp=' PICZZZ9
    DELIMITED BY SIZE INTO W-TEXT END-STRING.
    PERFORM SCRNLG2.
    MOVE CANDVER OF BINCONRS TO PICZZZ9.
    MOVE SPACES TO W-TEXT.
    STRING ' CandName=' CANDNAME OF BINCONRS
    ' CandVer=' PICZZZ9 ' CResGrp=' CANDRGRP OF BINCONRS
    ' CResAss=' CANDRASG OF BINCONRS ' CResDes='
    CANDRDSC OF BINCONRS
    DELIMITED BY SIZE INTO W-TEXT END-STRING.
    PERFORM SCRNLG2.
    MOVE CANDUSAGE OF BINCONRS TO CODEV.
    MOVE 'BINCONRS' TO CHR8.
    MOVE 'CANDUSAGE' TO CHR12.
    PERFORM XCV2CH
    MOVE CHARV TO CHAR6.
    MOVE CANDTYPE OF BINCONRS TO CODEV.
    MOVE 'BINCONRS' TO CHR8.
    MOVE 'CANDTYPE' TO CHR12.
    PERFORM XCV2CH
    MOVE CHARV TO CHAR12.
    MOVE CANDASGOVR OF BINCONRS TO CODEV.
    MOVE 'BINCONRS' TO CHR8.
    MOVE 'CANDASGOVR' TO CHR12.
    PERFORM XCV2CH
    MOVE SPACES TO W-TEXT.
    STRING ' CandUsa=' CHAR6
    ' CandSGrp=' CANDSGRP OF BINCONRS
    ' CandSTyp=' CHAR12 ' CandAss0=' CHARV
    DELIMITED BY SIZE INTO W-TEXT END-STRING.
    PERFORM SCRNLG2.
    MOVE EXISTVER OF BINCONRS TO PICZZZ9.
    MOVE EXISTUSAGE OF BINCONRS TO CODEV.
    MOVE 'BINCONRS' TO CHR8.
    MOVE 'EXISTUSAGE' TO CHR12.
    PERFORM XCV2CH
    MOVE SPACES TO W-TEXT.
    STRING ' ExistName=' EXISTNAME OF BINCONRS
    ' ExistVer=' PICZZZ9 ' EResGrp=' EXISTRGRP OF
    BINCONRS ' EResAss=' EXISTRASG OF BINCONRS
    ' EResDes=' EXISTRDSC OF BINCONRS ' ExistUsa=' CHARV
    DELIMITED BY SIZE INTO W-TEXT END-STRING.
    PERFORM SCRNLG2.
    MOVE EXISTTYPE OF BINCONRS TO CODEV.
    MOVE 'BINCONRS' TO CHR8.
    MOVE 'EXISTTYPE' TO CHR12.
    PERFORM XCV2CH
    MOVE CHARV TO CHAR12.
    MOVE EXISTASGOVR OF BINCONRS TO CODEV.
    MOVE 'BINCONRS' TO CHR8.
    MOVE 'EXISTASGOVR' TO CHR12.
    PERFORM XCV2CH
    MOVE SPACES TO W-TEXT.
    STRING ' ExistSGrp=' EXISTSGRP OF BINCONRS
    ' ExistSTyp=' CHAR12 ' ExistAss0=' CHARV
    DELIMITED BY SIZE INTO W-TEXT END-STRING.
    PERFORM SCRNLG2.
    EXIT.

```

```

*****
* This subroutine displays BINCONSC information *
*****
DISPBISC.
  MOVE W-BUFFER TO BINSTERR.
  MOVE ERROP OF BINCONSC TO PICZZZ9.
  MOVE ERRCODE OF BINCONSC TO PYCZZZ9.
  MOVE SPACES TO W-TEXT.
  STRING 'CMAS=' CMASNAME OF BINCONSC ' Plex='
  PLEXNAME OF BINCONSC ' EOp=' PICZZZ9 ' ECode='
  PYCZZZ9 ' TScope=' TARGSCOPE OF BINCONSC
  ' TAssgn=' TARGRASG OF BINCONSC
  DELIMITED BY SIZE INTO W-TEXT END-STRING.
  PERFORM SCRNLG2.
  MOVE SPACES TO W-TEXT.
  STRING ' TDesc=' TARGDSC OF BINCONSC ' RScope='
  RELSCOPE OF BINCONSC ' RAssgn=' RELRASG OF BINCONSC
  ' RDesc=' RELRDSC OF BINCONSC ' CSys=' CICSNAME OF
  BINCONSC
  DELIMITED BY SIZE INTO W-TEXT END-STRING.
  PERFORM SCRNLG2.
  EXIT.

```

```

*****
* This subroutine converts coded value to character string *
*****
XCV2CH.
* Use new thread for TRANSLATE
  EXEC CPSM CONNECT
      VERSION('0140')
      THREAD(W-FBTTKN)
      RESPONSE(W-RESPONSE)
      REASON(W-REASON)
  END-EXEC.

* Translate internal coded value to character value
  EXEC CPSM TRANSLATE
      OBJECT(CHR8)
      ATTRIBUTE(CHR12)
      FROMCV(CODEV) TOCHAR(CHARV)
      THREAD(W-FBTTKN)
      RESPONSE(W-RESPONSE)
      REASON(W-REASON)
  END-EXEC.
  EXIT.

```

```

*-----*

```

sample program EYULAPI4

```
*      PROCESSING FOR API FAILURES.                                     *
*-----*
NO-CONNECT.
  MOVE 'ERROR CONNECTING TO API.' TO W-MSG-TEXT.
  GO TO SCRNLG.
NO-CREATE.
  MOVE 'ERROR CREATING DEFINITION.' TO W-MSG-TEXT.
  GO TO SCRNLG.
NO-GET.
  MOVE 'ERROR GETTING RESOURCE TABLE.' TO W-MSG-TEXT.
  GO TO SCRNLG.
NO-INSTALL.
  MOVE 'ERROR INSTALLING RESULT SET.' TO W-MSG-TEXT.
  GO TO SCRNLG.
NO-TRANSLATE.
  MOVE 'ERROR TRANSLATING ATTRIBUTE.' TO W-MSG-TEXT.
  GO TO SCRNLG.
SCRNLG.
*  DISPLAY W-MSG-TEXT.
  EXEC CICS SEND FROM(W-MSG-TEXT) LENGTH(81) WAIT END-EXEC.
  MOVE W-RESPONSE TO PICZZZ9A.
  MOVE W-REASON TO PICZZZ9B.
  STRING 'RESPONSE=' DELIMITED BY SIZE
         PICZZZ9A DELIMITED BY SIZE
         ' REASON= ' DELIMITED BY SIZE
         PICZZZ9B DELIMITED BY SIZE
         INTO W-MSG-TEXT.
SCRNLG2.
*  DISPLAY W-MSG-TEXT.
  EXEC CICS SEND FROM(W-MSG-TEXT) LENGTH(81) WAIT END-EXEC.

  ENDIT.
*-----*
*  TERMINATE API CONNECTION.                                         *
*-----*
  EXEC CPSM TERMINATE RESPONSE(W-RESPONSE) REASON(W-REASON)
  END-EXEC.
  EXEC CICS RETURN END-EXEC.
*  GOBACK
  EXIT.
EYULAPI4-END.
```

The COBOL version of EYUxAPI4 is written for the CICS environment and can be converted to run in the MVS/ESA batch environment by commenting the EXEC CICS SEND commands, and uncommenting the preceding language specific output statement.

Glossary

This glossary defines CICSplex SM terms and abbreviations used in this book with other than their everyday meaning. Terms that are defined in the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994, are not defined here unless CICSplex SM usage is different from the meaning given there.

If you cannot find the definition you need, refer to the *Dictionary of Computing* or the *CICSplex SM Master Index*, SC33-1812.

A

action command. A CICSplex SM command that affects one or more of the resources represented in a view. Action commands can be issued from either the COMMAND field in the control area of the information display panel or the line command field in a displayed view. Valid action commands are listed with the description of each view. See also *overtyping field*.

action definition (ACTNDEF). In real-time analysis, a definition of the type of external notification that is to be issued when the conditions identified in an analysis definition are true.

activity. See *BTS activity*.

adjacent CMAS. A CICSplex SM address space (CMAS) that is connected to the local CMAS via a direct CMAS-to-CMAS link. Contrast with *indirect CMAS*. See also *local CMAS*.

alter expression. A character string that defines the changes to be made to a resource attribute. An alter expression is made up of one or more attribute expressions.

alternate window. A window to which the results of a hyperlink can be directed. By default, the results of a hyperlink are displayed in the same window from which the hyperlink is initiated. Contrast with *current window*.

alternate window (ALT WIN) field. In the control area of an information display panel, the field in which you can specify an alternate window to receive the results of a hyperlink.

analysis definition. In real-time analysis, a definition of the evaluations to be performed on specified CICS resources, the intervals at which those evaluations are to be performed, and the actions to be taken when a notifiable condition occurs.

analysis group. In real-time analysis, a group of one or more analysis definitions, status definitions, or both.

Analysis definitions and status definitions must belong to an analysis group if they are to be installed automatically in a CICS system when that system starts.

analysis point monitoring (APM). In real-time analysis, resource monitoring across multiple CICS systems within a CICSplex that results in a single notification of a condition, rather than one notification for each system. Contrast with *MAS resource monitoring*.

analysis point specification. In real-time analysis, a specification that identifies the CMASs that are to be responsible for analysis point monitoring.

analysis specification. In real-time analysis, a specification that establishes system availability monitoring or MAS resource monitoring within a group of CICS systems.

AOR. Application-owning region.

API. Application programming interface

APM. Analysis point monitoring.

application-owning region (AOR). In a CICSplex configuration, a CICS region devoted to running applications. For dynamic routing, the terms *requesting region*, *routing region*, and *target region* are used instead of AOR to signify the role of the region in the dynamic routing request.

ARM. Automatic restart manager.

ASU. Automatic screen update.

attribute. See *resource attribute*, *resource table attribute*.

attribute expression. A reference to a resource table attribute and, in some cases, its value. Attribute expressions are used to build filter expressions, modification expressions, and order expressions.

attribute value. The data currently associated with a resource table attribute. For example, the file attribute OPENSTATUS might have a value of CLOSED.

automatic restart manager (ARM). A recovery function of MVS/ESA 5.2 that provides improved availability for batch jobs and started tasks by restarting them automatically if they end unexpectedly. The affected batch job or started task can be restarted on the same system or on a different one, if the system itself has failed.

automatic screen update (ASU). A CICSplex SM facility that automatically updates the data in all unlocked windows at user-defined intervals. See also *automatic screen update interval*.

automatic screen update interval. The time interval between one automatic screen update and the next. This interval can be set in the CICSplex SM user profile or when the ASU facility is turned on. See also *automatic screen update (ASU)*.

B

BAS. Business Application Services

batched repository-update facility. A CICSplex SM facility, invoked from the CICSplex SM end user interface, for the bulk application of CICSplex SM definitions to a CMAS data repository.

BTS. CICS business transaction services

BTS activity. One part of a process managed by CICS BTS. Typically, an activity is part of a *business transaction*.

BTS process. A collection of more than one CICS BTS *activities*. Typically, a process is an instance of a *business transaction*.

BTS set. See CICS system group

business application. Any set of CICS resources that represent a meaningful entity to an enterprise or a user (such as, Payroll).

Business Application Services (BAS). The component of CICSplex SM that provides the ability to define and manage business applications in terms of their CICS resources and associated CICS systems. BAS provides a central definition repository for CICS systems, complete with installation facilities and the ability to restrict a CICSplex SM request to those resources defined as being part of the business application. See also *business application, scope*.

business transaction. A self-contained business function, for example, the booking of an airline ticket.

C

CAS. Coordinating address space.

CBIPO. Custom-built installation process offering.

CBPDO. Custom-built product delivery offering.

CEDA. A CICS transaction that defines resources online. Using CEDA, you can update both the CICS

system definition data set (CSD) and the running CICS system.

CICS Business Transaction Services (BTS). A CICS domain that supports an application programming interface (API) and services that simplify the development of *business transactions*.

CICS system. The entire collection of hardware and software required by CICS. In CICSplex SM topology, a definition referring to a CICS system that is to be managed by CICSplex SM. See also *CICSplex, CICS system group*.

CICS system group. A set of CICS systems within a CICSplex that can be managed as a single entity. In CICSplex SM topology, the user-defined name, description, and content information for a CICS system group. A CICS system group can be made up of CICS systems or other CICS system groups. In CICS business transaction services (BTS), a BTS set, that is the set of CICS regions across which BTS processes and activities may execute. See also *CICSplex, CICS system*.

CICSplex. A CICS complex. A CICSplex consists of two or more CICS regions that are linked using CICS intercommunication facilities. The links can be either intersystem communication (ISC) or interregion communication (IRC) links, but within a CICSplex are more commonly IRC. Typically, a CICSplex has at least one terminal-owning region (TOR), more than one application-owning region (AOR), and may have one or more regions that own the resources being accessed by the AORs. In CICSplex SM, a management domain. The largest set of CICS regions, or CICS systems, to be manipulated by CICSplex SM as a single entity. CICS systems in a CICSplex being managed by CICSplex SM do not need to be connected to each other. See also *CICS system, CICS system group*.

CICSplex SM. IBM CICSplex System Manager.

CICSplex SM address space (CMAS). A CICSplex SM component that is responsible for managing CICSplexes. A CMAS provides the single-system image for a CICSplex by serving as the interface to other CICSplexes and external programs. There must be at least one CMAS in each MVS image on which you are running CICSplex SM. A single CMAS can manage CICS systems within one or more CICSplexes. See also *coordinating address space (CAS), managed application system (MAS)*.

CICSplex SM token. Unique, 4-byte values that CICSplex SM assigns to various elements in the API environment. Token values are used by CICSplex SM to correlate the results of certain API operations with subsequent requests.

client program. In dynamic routing, the application program, running in the *requesting region*, that issues a remote link request.

CMAS. CICSplex SM address space.

CMAS link. A communications link between one CICSplex SM address space (CMAS) and another CMAS or a remote managed application system (remote MAS). CMAS links are defined when CICSplex SM is configured.

CODB. A CICSplex SM transaction for interactive, system-level debugging of CMASs and of CICS/ESA, CICS/MVS, and CICS/VSE MASs. CODB must be used only at the request of customer support personnel.

COD0. A CICSplex SM transaction for interactive, method-level debugging of CMASs and of CICS/ESA, CICS/MVS, CICS/VSE, and CICS for OS/2 MASs. COD0 must be used only at the request of customer support personnel.

COLU. A CICSplex SM transaction for generating reports about CMAS and local MAS components. COLU must be used only at the request of customer support personnel.

COMMAND field. In the control area of an information display panel, the field that accepts CICSplex SM, ISPF, and TSO commands. Contrast with *option field*.

command-level interface. A CICSplex SM API interface that uses the CICS translator to translate EXEC CPSM statements into an appropriate sequence of instructions in the source language.

Common Services. A component of CICSplex SM that provides commonly requested services (such as GETMAIN, FREEMAIN, POST, and WAIT processing) to other CICSplex SM components.

communication area (COMMAREA). A CICS area that is used to pass data between tasks that communicate with a given terminal. The area can also be used to pass data between programs within a task.

Communications. A component of CICSplex SM that provides all services for implementing CMAS-to-CMAS and CMAS-to-MAS communication.

context. A named part of the CICSplex SM environment that is currently being acted upon by CICSplex SM. For configuration tasks, the context is a CICSplex SM address space (CMAS); for all other tasks, it is a CICSplex. See also *scope*.

control area. The top three lines of an information display panel, containing the panel title, the screen update time, the short message area, the COMMAND

and SCROLL fields, and the current window (CUR WIN) and alternate window (ALT WIN) fields.

coordinating address space (CAS). An MVS subsystem that provides ISPF end-user access to the CICSplex to be accessed. See also *CICSplex SM address space, managed application system (MAS)*.

coordinating address space subsystem ID. Identifies the coordinating address space (CAS) which can be up to 4 characters, to be connected to when issuing CICSplex SM requests. The name of the CAS is installation-dependent, and is defined in the CICSplex SM user profile.

cross-system coupling facility (XCF). XCF is a component of MVS that provides functions to support cooperation between authorized programs running within a sysplex.

current window. The window to which the results of all commands issued in the COMMAND field are directed, unless otherwise requested. Contrast with *alternate window*.

current window (CUR WIN) field. In the control area of an information display panel, the field that contains the window number of the current window. You can change the number in this field to establish a new current window.

custom-built installation process offering (CBIPO). A product that simplifies the ordering, installation, and service of MVS system control programs and licensed programs by providing them with current updates and corrections to the software that is already integrated.

custom-built product delivery offering (CBPDO). A customized package of both products and service, or of service only, for MVS system control programs and licensed programs.

D

Data Cache Manager. A component of CICSplex SM that manages logical cache storage for use by other CICSplex SM components.

data repository. In CICSplex SM, the VSAM data set that stores administrative data, such as topology and monitor definitions, for a CICSplex SM address space (CMAS).

Data Repository. A component of CICSplex SM that provides methods for creating, accessing, updating, and deleting data in the CICSplex SM data repository. See also *Managed Object Services*.

Database Control (DBCTL). An IMS/ESA facility providing an interface between CICS/ESA and IMS/ESA

that allows access to IMS DL/I full-function databases and to data-entry databases (DEDBs) from one or more CICS/ESA systems.

Database 2 (DB2). An IBM licensed program. DB2 is a full-function relational database management system that presents a data structure as a table consisting of a number of rows (or records) and a number of columns.

DBCTL. Database Control.

DB2. Database 2.

derived field. On a monitor view, a field whose value does not come directly from CICS or CICSplex SM data, but is calculated based on the values in other fields. See also *derived value*.

derived value. A rate, average, or percentage that results from CICSplex SM processing of CICS statistics.

display area. On an information display panel, the area where windows can be opened to display data. The display area appears below the control area. The bottom two lines of the display area can be used to display the PF key assignments in effect for a CICSplex SM session.

display attributes. A CICSplex SM user profile option that controls the appearance of the window information line, field headings, and threshold values in a view.

display command. A CICSplex SM command that extends the ISPF interface to create and control a multiwindow environment.

distributed program link (DPL). Function of CICS intersystem communication that enables CICS to ship LINK requests between CICS regions.

distributed routing program (DSRTPGM). A CICS-supplied user-replaceable program that can be used to dynamically route:

- CICS BTS processes and activities
- Transactions started by non-terminal related EXEC CICS START commands

DPL. Distributed program link.

DTR. Dynamic transaction routing.

dynamic routing. The automatic routing of a transaction or program, at the time it is initiated, from a requesting region to a suitable target region. Routing terminal data to an alternative transaction at the time the transaction is invoked. To do this, CICS allows the

dynamic routing program to intercept the terminal data and redirect it to any system and transaction it chooses. See also dynamic routing program (EYU9XLOP)

dynamic routing program (EYU9XLOP). A user-replaceable CICS program that selects dynamically both the system to which a routing request is to be sent and the transaction's remote name. The alternative to using this program is to make these selections when a remote transaction is defined to CICS (static routing). See also *static routing*

dynamic transaction routing (DTR). The automatic routing of a transaction, at the time it is initiated, from a transaction-owning region (TOR) to a suitable application-owning region (AOR).

E

Environment Services System Services (ESSS). A component of CICSplex SM that implements the formal MVS/ESA subsystem functions required by the product. ESSS provides cross-memory services, data space management, connection services, and lock management. An ESSS system address space is created at CICSplex SM initialization and remains in the MVS image for the life of the IPL.

ESSS. Environment Services System Services.

evaluation definition. In real-time analysis, a definition of the resources that are to be sampled. When the result of an evaluation is true, an associated analysis definition is used to determine whether a notifiable condition has occurred.

event. A significant occurrence within the CICSplex or system for which the user has requested notification. For example, the end of processing, a subsystem failure, or any unusual condition in the system could be defined by a user as an event.

event notification. A CICSplex SM notification of a significant occurrence within a CICSplex or CICS system.

extended diagnostic mode (XDM). A CICSplex SM online internal diagnostic facility. XDM provides no information about resources managed by CICSplex SM, and should be turned on only at the request of IBM customer support personnel. XDM can be turned on and off in the CICSplex SM user profile.

external notification. In RTA, an event notification, generic alert, or operator message issued when a notifiable condition occurs.

F

file-owning region. In a CICSplex configuration, a CICS system devoted to managing CICS file access.

filter expression. A character string that consists of logical expressions to be used in filtering resource table records. A filter expression is made up of one or more attribute expressions.

FOR. File-owning region.

form. The way in which data obtained from a query is presented in a view. See also *query*, *view*.

G

generic alert. A Systems Network Architecture (SNA) Network Management Vector that enables a product to signal a problem to the network. CICSplex SM uses generic alerts as part of its interface to NetView.

GMFHS. Graphic Monitor Facility host subsystem.

goal algorithm. In CICSplex SM's workload balancing, an algorithm used to select an AOR to process a dynamic transaction. Using the goal algorithm, CICSplex SM selects the AOR that is the least affected by conditions such as short-on-storage, SYSDUMP, and TRANDUMP; is the least likely to cause the transaction to abend; and is most likely to enable the transaction to meet response-time goals set for it using the Workload Manager component of MVS/ESA SP 5.1. Contrast with *queue algorithm*.

Graphic Monitor Facility host subsystem. A NetView feature that manages configuration and status updates for non-SNA resources.

H

hyperlink. A direct connection between the data in one CICSplex SM view and a view containing related information. For example, from a view that lists multiple CICS resources, there may be a hyperlink to a detailed view for one of the resources. To use a hyperlink, place the cursor in the data portion of a hyperlink field and press Enter.

hyperlink field. On a CICSplex SM view, a field for which a hyperlink is defined. The headings of hyperlink

fields are shown in high intensity or color, depending on the terminal type.

I

IBM CICSplex System Manager for MVS/ESA (CICSplex SM). An IBM CICS system-management product that provides a single-system image and a single point of control for one or more CICSplexes that can be installed on heterogeneous operating systems.

indirect CMAS. A CICSplex SM address space (CMAS) that the local CMAS can communicate with via an adjacent CMAS. There is no direct CMAS-to-CMAS link between the local CMAS and an indirect CMAS. Contrast with *adjacent CMAS*. See also *local CMAS*.

information display panel. The panel that supports the CICSplex SM window environment. It consists of a control area and a display area. CICSplex SM views are displayed in windows within the display area of this panel.

information display parameters. A CICSplex SM user profile option that defines the initial screen configuration, how frequently the screen will be updated by ASU, and how long a window will wait for command processing to complete before timing out.

installation verification procedure (IVP). A procedure distributed with a system that tests the newly generated system to verify that the basic facilities of the system are functioning correctly.

interregion communication. Synonym for *multiregion operation*.

intersystem communication (ISC). Communication between separate systems by means of SNA networking facilities or by means of the application-to-application facilities of an SNA access method.

intertransaction affinity. A relationship between CICS transactions, usually the result of the ways in which information is passed between those transactions, that requires them to execute in the same CICS region. Intertransaction affinity imposes restrictions on the dynamic routing of transactions.

IRC. Interregion communication.

ISC. Intersystem communication.

IVP. Installation verification procedure.

K

Kernel Linkage. A component of CICSplex SM that is responsible for building data structures and managing the interfaces between the other CICSplex SM components. The environment built by Kernel Linkage is known as the method call environment.

L

line command field. In a CICSplex SM view, the 3 character field, to the left of the data, that accepts action commands.

local CMAS. The CICSplex SM address space (CMAS) that a user identifies as the current context when performing CMAS configuration tasks.

local MAS. A managed application system (MAS) that resides in the same MVS image as the CICSplex SM address space (CMAS) that controls it and that uses the Environment Services System Services (ESSS) to communicate with the CMAS.

logical scope. A set of logically related CICS resources that are identified in a CICSplex SM resource description. A logical scope can be used to qualify the context of a CICSplex SM request.

M

maintenance point. A CICSplex SM address space (CMAS) that is responsible for maintaining CICSplex SM definitions in its data repository and distributing them to other CMASs involved in the management of a CICSplex. See also *data repository*.

Major object descriptor block (MODB). In CICSplex SM, a control structure built by Kernel Linkage during initialization of a CICSplex SM component that contains a directory of all methods that make up that component. The structure of the MODB is the same for all components.

Major object environment block (MOEB). In CICSplex SM, a control structure built by Kernel Linkage during initialization of a CICSplex SM component and pointed to by the MODB. The MOEB stores information critical to a CICSplex SM component and anchors data used by the component. The structure of the MOEB is unique to the component it supports.

MAL. Message argument list.

managed application system (MAS). A CICS system that is being managed by CICSplex SM. See *local MAS*, *remote MAS*.

managed object. A CICSplex SM-managed CICS resource or a CICSplex SM definition represented by a resource table. A view is based on a single managed object.

Managed Object Services. A subcomponent of the Data Repository component of CICSplex SM that translates a request for data (from real-time analysis, for example) into the method calls required to obtain the data.

MAS. Managed application system.

MAS agent. A CICSplex SM component that acts within a CICS system to provide monitoring and data collection for the CICSplex SM address space (CMAS). The level of service provided by a MAS agent depends on the level of CICS the system is running under and whether it is a local or remote MAS. See also *CICSplex SM address space (CMAS)*, *local MAS*, *remote MAS*.

MAS resource monitoring (MRM). In real-time analysis, resource monitoring at the CICS system level; it results in one notification of a condition for each system in which it occurs. If the same condition occurs in three CICS systems where MAS resource monitoring is active, three notifications are issued. Contrast with *analysis point monitoring*.

Message argument list (MAL). In CICSplex SM, a data structure passed between methods using Kernel Linkage method call services.

message line. On an information display panel, the line in the control area where a long message appears when the HELP command is issued in response to a short message. The message line temporarily overlays the CURR WIN and ALT WIN fields.

Message Services. A component of CICSplex SM that provides services for building and issuing MVS/ESA console messages to other CICSplex SM components.

meta-data. Internal data that describes the structure and characteristics of CICSplex SM managed objects.

method. (Action.) An application programming interface (API) instruction that resolves into an EXEC CICS command, issued against one or more resources in one or more CICS systems, within the current context and scope.

method. In CICSplex SM, one of the programs that make up a CICSplex SM component. See also *message argument list (MAL)*.

mirror transaction. CICS transaction that recreates a request that is function shipped from one system to another, issues the request on the second system, and passes the acquired data back to the first system.

MODB. Major object descriptor block.

modification expression. A character string that defines the changes to be made to a resource attribute. A modification expression is made up of one or more attribute expressions.

MOEB. Major object environment block.

monitor definition. A user-defined statement of the specific resource occurrences (such as the program named PAYROLL) to be monitored by CICSplex SM. A monitor definition can either be linked to a monitor specification as part of a monitor group or be installed directly into an active CICS system. See also *monitor group*, *monitor specification*.

monitor group. A user-defined set of CICSplex SM monitor definitions that can either be linked to a monitor specification for automatic installation or be installed directly into an active CICS system. See also *monitor definition*, *monitor specification*.

monitor interval. The number of minutes that are to elapse before the statistics counters containing accumulated resource monitoring data are automatically reset. This value is part of a CICSplex definition and affects all of the CICS systems and CICS system groups associated with that CICSplex. See also *period definition*, *sample interval*.

monitor specification. A user-defined statement of the types of resources (such as programs) to be monitored by CICSplex SM and how often data should be collected. A monitor specification is associated with a CICS system and is automatically installed each time the CICS system starts up. See also *monitor definition*, *monitor group*.

Monitoring Services. A component of CICSplex SM that is responsible for monitoring resources within a CICS system and making the collected data available to other CICSplex SM components.

MRM. MAS resource monitoring.

MRO. Multiregion operation.

MSM. MultiSystem Manager.

multiregion operation (MRO). Communication between CICS systems without the use of SNA network facilities. Synonymous with *interregion communication*.

MultiSystem Manager. An object-oriented, graphical systems management application that runs under NetView for MVS.

MVS image. A single instance of the MVS operating system.

MVS system. An MVS image together with its associated hardware.

N

NetView. An IBM network management product that can provide rapid notification of events and automated operations. CICSplex SM can be set up to send generic alerts to NetView as part of its event processing capabilities.

NetView Graphic Monitor Facility (NGMF). A function of the NetView program that provides the network operator with a graphic topological presentation of a network controlled by the NetView program and that allows the operator to manage the network interactively.

NetView program. An IBM licensed program used to monitor and manage a network and to diagnose network problems.

NGMF. NetView Graphic Monitor Facility.

notification. A message that is generated asynchronously by a CICSplex SM managed object to describe an event related to the object.

O

option field. On a CICSplex SM menu, the field in which you can specify an option number or letter. Contrast with *command field*.

order expression. A character string that defines either the attributes to be used in sorting resource table records, or the attributes to be included in a resource table view. An order expression is made up of one or more attribute expressions.

override expression. A character string that defines the changes to be made to a resource attribute. An override expression is made up of one or more attribute expressions.

overtyping field. On a CICSplex SM view, a field containing a value that can be changed by typing a new value directly into the field. Values that can be overtyped are shown in high intensity or color, depending on the terminal type. Acceptable values for overtyping fields are listed with the description of each view. See also *action command*.

P

parameter expression. A character string that defines the parameters required for an action to complete or a definition to be processed.

parameter repository. In CICSplex SM, a data set that stores cross-system communication definitions that allow one coordinating address space (CAS) to communicate with other CASs.

period definition. A user-defined range of hours and minutes and the time zone to which that range applies. A period definition is used to indicate when an action, such as resource monitoring, is to occur. See also *monitor interval*, *sample interval*.

PlexManager. A service utility that can be used to manage the communication connections between multiple coordinating address spaces (CASs) and between a CAS and its associated CICSplex SM address spaces (CMASs) and CICSplexes.

process. See *BTS process*

processing thread. A connection between an application program and the CICSplex SM API. A program can establish multiple processing threads, but each one is considered a unique API user; no resources can be shared across the boundary of a thread.

pseudoconversation. A CICS application designed to appear to the user as a continuous conversation, but that consists internally of multiple separate tasks.

Q

query. A request for specific data that is generated by a view command. See also *form*, *view*.

queue algorithm. In CICSplex SM's workload balancing, an algorithm used to select an AOR to process a dynamic transaction. Using the queue algorithm, CICSplex SM selects the AOR that has the shortest queue of transactions (normalized to MAXTASKs) waiting to be processed; is the least affected by conditions such as short-on-storage, SYSDUMP, and TRANDUMP; and is the least likely to cause the transaction to abend. Contrast with *goal algorithm*.

Queue Manager. A component of CICSplex SM that creates and manages queues of data in a cache that is shared by a CMAS and its local MASs.

R

RACF. Resource Access Control Facility.

real-time analysis (RTA). A component of CICSplex SM that is responsible for monitoring the status of a CICS system or resource against its desired status, and issuing one or more external notifications when deviations occur.

record pointer. An internal indicator of the next resource table record to be processed in a result set.

related scope. A CICS system where resources defined to CICSplex SM as remote should be assigned and, optionally, installed as local resources. See also *target scope*.

remote MAS. A managed application system (MAS) that uses MRO or LU 6.2 to communicate with the CICSplex SM address space (CMAS) that controls it. A remote MAS may or may not reside in the same MVS image as the CMAS that controls it.

requesting region. The region in which a dynamic routing request originates. For dynamic transaction routing and inbound client dynamic program link requests, this is typically a TOR; for dynamic START requests and peer-to-peer dynamic program link requests, this is typically an AOR.

resource. Any physical or logical item in a CICS system, such as a transient data queue, a buffer pool, a file, a program, or a transaction.

Resource Access Control Facility (RACF). An IBM licensed program that provides for access control by identifying and verifying the users to the system, authorizing access to protected resources, logging any detected unauthorized attempts to enter the system, and logging the detected accesses to protected resources.

resource assignment. A user-defined statement that selects resource definitions to be assigned to CICS systems and, optionally, specifies resource attributes to override those definitions. A resource assignment applies to a single resource type and must be associated with a resource description. See also *resource definition*, *resource description*.

resource attribute. A characteristic of a CICS resource, such as the size of a buffer pool.

resource definition. In CICSplex SM, a user-defined statement of the physical and operational characteristics of a CICS resource. Resource definitions can be associated with resource descriptions as part of a resource group. See also *resource description*, *resource group*.

resource description. A user-defined set of CICSplex SM resource definitions that can be automatically installed in CICS systems and named as a logical scope for CICSplex SM requests. Resource descriptions represent the largest set of CICS resources that can be managed by CICSplex SM as a single entity. A resource description can be associated with one or more resource assignments. See also *logical scope*, *resource assignment*, *resource definition*.

resource group. A user-defined set of CICSplex SM resource definitions. A resource group can be associated with resource descriptions either directly or by means of resource assignments. See also *resource assignment*, *resource definition*, *resource description*.

Resource Object Data Manager (RODM). A component of the NetView program that operates as a cache manager and that supports automation applications. RODM provides an in-memory cache for maintaining real-time data in an address space that is accessible by multiple applications.

resource table. The external representation of a CICSplex SM managed object. A resource table defines all the attributes, or characteristics, of a managed object.

resource table attribute. A characteristic of a CICSplex SM managed object, as represented by a field in a resource table.

resource type. A group of related resources, such as files.

result set. A logical group of resource table records that can be accessed, reviewed, and manipulated by an API program.

retention period. For a monitored CICS system, the period of time for which monitor data is retained after the system becomes inactive. If a system is being monitored, becomes inactive, and remains inactive beyond the specified retention period, the monitor data is discarded. If the system becomes active before the retention period expires, the monitor data gathered before the system became inactive is retained, and monitoring continues.

RODM. Resource Object Data Manager.

routing region. The region in which the decision is made as to which is the most suitable target region for a dynamic routing request. For dynamic transaction routing, dynamic START requests, and inbound client dynamic program link requests, this is typically a TOR; for dynamic peer-to-peer program link requests, this is typically an AOR.

RTA. real-time analysis.

run-time Interface. A CICSplex SM API interface that accepts commands in the form of text strings and generates the appropriate API calls. The run-time interface supports programs written as REXX EXECs.

S

SAM. System availability monitoring.

sample interval. The duration, in seconds, between occurrences of data collection for a specific resource type. See also *monitor interval*, *period definition*, *resource type*.

scope. A named part of the CICSplex SM environment that qualifies the context of a CICSplex SM request. The scope can be the CICSplex itself, a CICS system, a CICS system group, or any set of CICS resources that are defined as a logical scope in a CICSplex SM resource description. For configuration tasks, where the context is a CICSplex SM address space (CMAS), the scope is ignored. When you are applying security, scope must be a single CICS system or CICSplex. It cannot be a CICS system group or any combination of individual CICSplexes or CICS systems. See also *context*, *logical scope*.

screen configuration. A user-defined, named layout of windows and the context, scope, view, and sort order associated with each. The initial configuration to be displayed when CICSplex SM is accessed can be identified on the user profile.

screen repository. In CICSplex SM, a data set that stores screen configuration definitions created by the SAVESCR display command. See also *screen configuration*.

selection list. In CICSplex SM, a data set that stores cross-system communication definitions that allow one coordinating address space (CAS) to communicate with other CASs.

selection list. A list of named items, such as views or screen configurations, from which one can be selected.

server program. In dynamic routing, the application program specified on the link request, and which is executed in the *target region*.

service point. One of the combinations of products and contexts that is known to the coordinating address space (CAS) to which you are connected. See also *context*.

session control parameters. A CICSplex SM user profile option that sets the coordinating address space (CAS) subsystem ID used for accessing CICSplex SM views and controls the extended diagnostic mode (XDM).

short message area. In the control area of an information display panel, that part of the title line that displays short messages.

single point of control. The ability to access and manage all CICS systems and their resources in a CICSplex from a single terminal or user session.

single system image. The collection and presentation of data about multiple CICS systems as though they were a single CICS system. In CICSplex SM, the single-system image is provided by the CICSplex SM address space (CMAS).

specification. See *analysis specification, monitor specification, workload specification.*

Starter Set. A part of CICSplex SM comprising sample CICSplex SM definitions and sample JCL. The Starter Set samples may be used as supplied for educational purposes. They may also be copied and adapted for the customer environment.

static routing. Non-dynamic routing. The routing request is routed to a predetermined system. Static transaction routing occurs when NO is specified in the Dynamic field in either the transaction definition or the program definition. In both cases, the request is routed to the system named in the Remote Sysid field.

status definition. In real-time analysis, a definition of a user-written program to be invoked at specified intervals to evaluate the status of a non-CICS resource.

summarized result set. A special type of result set that is produced by grouping, or summarizing, the resource table records in a result set. See also *result set.*

summary expression. A character string that consists of one or more summary options and the resource table attributes to which they apply. See also *summary option.*

summary option. A value that indicates how the attribute values in a resource table are to be summarized.

sysplex. A set of MVS systems communicating and cooperating with each other through specific multisystem hardware components and software services to process customer workloads.

system availability monitoring (SAM). In real-time analysis, the monitoring of CICS systems to determine whether: they are active during their defined hours of operation; they are experiencing a short-on-storage, SYSDUMP, TRANDUMP, MAXTASK, or STALL condition. If a CICS system becomes inactive or one of the specified conditions occurs, an external notification is issued.

system image. The representation of a program and its related data as it exists in main storage.

T

target region. The region selected from a set of target regions as the most suitable region in which to execute the work request. For all dynamic routing requests, this is typically an AOR.

target scope. A CICS system or CICS system group where resources defined to CICSplex SM should be assigned and, optionally, installed. See also *related scope.*

temporary maintenance point. A CICSplex SM address space (CMAS) that serves as the maintenance point when the identified maintenance point is unavailable. See also *maintenance point.*

terminal-owning region. In a CICSplex configuration, a CICS region devoted to managing the terminal network. For dynamic routing, the terms *requesting region* and *routing region* are used instead of TOR to signify the role of the region in the dynamic routing request.

thread. See *processing thread.*

time-period definition. A user-defined range of hours and minutes, and the time zone to which that range applies. A time-period definition is used to indicate when an action, such as resource monitoring, is to occur.

token. See *CICSplex SM token, user token.*

topology. An inventory of CICS and CICSplex SM resources, and a map of their relationships. CICSplex SM supports the definition of resource and system topology.

topology definition. A named subset of CICS and CICSplex SM resources. Topology definitions are user-created and can include CICSplexes, CICS systems, and CICS system groups.

Topology Services. A component of CICSplex SM that is responsible for maintaining topology information about CICSplexes and resources, and making it available to other CICSplex SM components.

TOR. Terminal-owning region.

Trace Services. A component of CICSplex SM that provides other CICSplex SM components with the ability to write trace records to the CICS trace table and trace data sets. Trace Services also writes trace

records created by a MAS to the trace table and data set of the managing CMAS.

transaction group. A user-defined, named set of transactions that determines the scope of workload balancing and the affinity relationships between transactions.

U

user token. Unique, 1- to 4-byte values that an API user can assign to asynchronous requests. User token values are not used by CICSplex SM; they are simply held until the request is complete and then returned to the user.

V

view. In the CICSplex SM API, a temporary, customized form of a resource table. A view can consist of some or all of the resource table attributes in any order. In the CICSplex SM ISPF end-user interface, a formatted display of selected data about CICS resources or CICSplex SM definitions. The data in a view is obtained from a query and can be presented in one or more forms. The data can be limited to a subset of CICSplex resources or definitions by establishing a context and scope.

view command. A CICSplex SM command that displays a view in a window of the display area. The name of the view displayed matches the name of the view command. See also *view*.

W

window. In CICSplex SM, a subdivision of the display area. The results of any CICSplex SM view or display command are directed to a single window, which is the current window by default. Contrast with *view*. See also *current window*, *alternate window*.

window identifier. On a window information line, the field that identifies the window. A window identifier consists of a one-character status code and a number in the range 1 through 20.

window information line. The top line of each window in the display area. It includes the window identifier, the name of the view displayed in the window, the context and scope in effect, the date and time when the view was last refreshed, and the product name.

window number. A number assigned by CICSplex SM to a window when it is opened. The window number is the second part of the window identifier on the window information line.

window status code. A one-character code that indicates whether a window is ready to receive commands, is busy processing commands, is not to be updated, or contains no data. It also indicates when an error has occurred in a window. The window status code is the first character of the window identifier on the window information line.

WLM. Workload Manager.

workload. The total number of transactions that a given CICSplex is intended to process in a specific period. For example, a workload could be expressed as a number of transactions per hour, or per day. In CICSplex SM, a named set of transactions and CICS systems, acting as requesting regions, routing regions, and target regions that form a single, dynamic entity.

workload balancing. The technique of balancing a workload across multiple target regions that are capable of processing the work.

workload definition. A user-defined statement of the transaction groups associated with a CICS system that is an AOR. A workload definition can either be linked to a workload specification as part of a workload group or be installed directly into an active workload. See also *workload group*, *workload specification*.

workload group. A user-defined set of CICSplex SM workload definitions that can either be linked to a workload specification for automatic installation or be installed directly into an active workload. See also *workload definition*, *workload specification*.

Workload Manager (WLM). A component of CICSplex SM that is responsible for managing the transaction workload in a CICSplex through the use of dynamic transaction routing.

workload separation. The technique of separating a workload into discrete parts, and allocating specific transactions to specific AORs.

workload specification. A user-defined statement that identifies a workload and a set of CICS systems acting as AORs. A workload specification also provides default management criteria for transactions that are not defined to CICSplex SM. It is associated with a CICS system that is a TOR and is automatically installed each time the CICS system starts up. See also *workload definition*, *workload group*.

X

XCF. Cross-system coupling facility of MVS/ESA.

XDM. Extended diagnostic mode

Index

A

- accessing API from REXX 91
- accessing CICSplex SM 2
- accessing resource tables from REXX 93
- actions, performing 37
- Assembler language programs
 - compiling 71
 - language considerations 68
 - link editing 73
 - run-time considerations 74
 - supported environments 2
 - translating 70
 - using resource table copy books 57
- asynchronous processing
 - overview 39
 - using ADDRESS 42
 - using LISTEN 41
 - using NOWAIT 41
 - using RECEIVE 43
 - using tokens 42
- ASYNCREQ records
 - description 41
 - retrieving 43
- attribute expression
 - in filter expression 20
 - in modification expression 36
- attributes, resource table
 - modifying 36
 - ordering 16
 - translating
 - in REXX program 94
- availability, CICS release 1

B

- BINCONRS resource table records 87, 88, 101
- BINCONSC resource table records 87, 88, 101
- BINSTERR resource table records 86, 102

C

- C programs
 - compiling 72
 - link editing 74
 - run-time considerations 74
 - running under NetView 69
 - supported environments 2
 - translating 71
 - using resource table copy books 65
- CHANGETIME attribute
 - description 16, 38
 - processing with REXX 95

- CICS definitions
 - description 14
 - working with 37
- CICS Global User exit programs 69
- CICS release availability 1
- CICS resources, managed
 - description 13
 - resource tables 15
- CICSplex SM API in status program 69
- CICSplex SM API in user-replaceable program 69
- CICSplex SM definitions
 - description 14
 - resource tables 15
 - working with 37
- CICSplex SM manager resources
 - description 14
 - resource tables 15
- CICSplex SM meta-data
 - description 15
 - resource tables 15
- CICSplex SM notifications
 - description 14
 - processing 41
 - resource tables 15
- CICSplex SM tokens 44
- CICSplex SM API task related user exit 53
- COBOL programs
 - compiling 72
 - link editing 74
 - run-time considerations 74
 - supported environments 2
 - translating 71
 - using resource table copy books 62
- command responses
 - testing for
 - using the command-level interface 80
 - using the run-time interface 80
 - types 77
- command-level interface
 - compiling a program 71
 - environment considerations 68
 - language considerations 68
 - link editing a program 73
 - run-time considerations 74
 - supported environments 1
 - translating a program 70
 - using resource table copy books 55
- compatibility of API programs
 - between environments 6
 - between releases 7
- compiling a command-level program 71

index

- CONNECT command
 - using 2
- connecting to CICSplex SM 2
- context
 - description 19
 - specifying on commands 19
- copy books, resource table 101
 - accessing 55
 - Assembler H 57
 - BINCONRS 87, 101
 - BINCONSC 88, 101
 - BINSTERR 86, 102
 - C/370 65
 - data characteristics 56
 - description 55
 - format 56
 - names and aliases 55
 - PL/I 58
 - VS COBOL II 62
- CREATETIME attribute
 - description 38
 - processing with REXX 95
- customizing resource table records 16

D

- definitions, CICS
 - description 14
 - working with 37
- definitions, CICSplex SM
 - description 14
 - resource tables 15
 - working with 37

E

- ECB field
 - description 42
- environment
 - compatibility 6
 - considerations 69
 - support 1
- error codes 101
- error handling
 - in REXX programs 97
 - using error result sets 86
 - using FEEDBACK data 81
 - using RESPONSE and REASON 77
- error result set
 - description 86
 - fields in FEEDBACK record 84
 - for BAS definitions 87, 88
 - for installing CICS resources 86
 - for updating CICS definitions 86
- event control block (ECB)
 - description 42

- event, listening for 41
- expression
 - attribute
 - in filter expression 20
 - in modification expression 36
 - filter 20
 - modification 36
 - order 17, 35
 - parameter 37, 38
 - summary 33
- EYU_attributes 16
- EYU_TRACE stem variable 98
- EYU9AR00 91
- EYU9AR01 91
- EYU9XESV security routine
 - considerations 5
- EYU9XLAP 53
- EYUAPI function
 - using 91, 92
- EYUINIT function
 - using 91
- EYUREAS function
 - using 80
- EYURESP function
 - using 80
- EYUTERM function
 - using 92
- EYUVALUE function
 - using for response and reason 80

F

- FEEDBACK command
 - using 81
- feedback records
 - availability 84
 - description 82
 - example 85
 - location 81
 - retrieving 81
- FETCH command
 - using 26
- filter
 - description 20
- filter expression
 - description 20
- filtering result set records 20
- function package, REXX 91

G

- GROUP command
 - using 32

L

- language considerations
 - Assembler H 68
 - PL/I 68
- link editing a command-level program 73
- LISTEN command
 - using 41
- listening for event 41
- LOCATE command
 - using 28
- locating a result set record 28

M

- managed CICS resources
 - description 13
 - resource tables 15
- managed object
 - modifying 35
 - selecting 19
 - types 13
- MARK command
 - using 29
- meta-data, CICSplex SM
 - description 15
 - resource tables 15
- migrating an API program 7
- modification expression
 - description 36
- modifying CICS definitions 38
- modifying CICSplex SM definitions 38
- modifying resource attributes 36

N

- NetView
 - considerations for C programs 69
 - languages supported 2
- notifications, CICSplex SM
 - description 14
 - processing 41
 - resource tables 15
- NOWAIT option, using 41

O

- objects, managed by CICSplex SM
 - modifying 35
 - selecting 19
 - types 13
- OBJSTAT records
 - description 26
 - in summarized result set 33
 - retrieving 26
- ORDER command
 - using 34

- order expression
 - description 17, 35
- ordering result set records 34

P

- parameter expression
 - for CICS definitions 30, 38
 - for CICSplex SM definitions 38
 - when performing an action 37
- performing actions 37
- PL/I programs
 - compiling 72
 - language considerations 68
 - link editing 74
 - run-time considerations 74
 - supported environments 2
 - translating 71
 - using resource table copy books 58
- programs, sample
 - descriptions 11
 - list of supplied 11
 - listings 103

R

- REASON option
 - using 77
- RECEIVE command
 - using 43
- record pointer, positioning 28
- release compatibility 7
- resource table
 - copy books 55
 - customizing 16
 - description 15
 - restricted attributes 16
 - translating attributes
 - in REXX program 94
 - using with command-level interface 55
 - using with REXX 93
 - view 16
- resource table copy books 101
 - accessing 55
 - Assembler H 57
 - BINCONRS 87, 101
 - BINCONSC 88, 101
 - BINSTERR 86, 102
 - C/370 65
 - data characteristics 56
 - description 55
 - format 56
 - names and aliases 55
 - PL/I 58
 - VS COBOL II 62

- RESPONSE option
 - using 77
- responses, command
 - testing for
 - using the command-level interface 80
 - using the run-time interface 80
 - types 77
- restricted resource table attributes 16
- result set
 - commands
 - overview 23
 - creating 23
 - description 23
 - positioning record pointer 28
 - records
 - customizing 16
 - filtering 20
 - locating 28
 - retrieving 26
 - sorting 34
 - summarizing 32
- result set, error
 - description 86
 - fields in FEEDBACK record 84
 - for BAS definitions 87, 88
 - for installing CICS resources 86
 - for updating CICS definitions 86
- retrieving ASYNCREQ records 43
- retrieving FEEDBACK records 81
- retrieving OBJSTAT records 26
- retrieving result set records 26
- REXX function package 91
- REXX run-time interface
 - accessing resource tables 93
 - EYU_TRACE data 98
 - function package 91
 - messages 98
 - run-time errors 98
 - STATUS values 98
 - supported environments 2
 - translation errors 97
 - using 91
- run-time considerations, command-level 74
- run-time errors, REXX 98

S

- sample programs
 - descriptions 11
 - list of supplied 11
 - listings 103
- scheduling a request 41
- scope
 - description 19
 - specifying on commands 19

- security
 - considerations 5
- selecting managed objects
 - using context and scope 19
 - using filter expressions 20
- sentinel field
 - description 42
- sorting result set records 34
- SPECIFY VIEW command
 - using 16
- status program
 - CICSplex SM API 69
- STATUS values, interpreting 98
- summarized result set
 - description 32
- summarizing result set records 32
- summary expression
 - description 33
- summary options
 - description 34
- supported environments 1

T

- task related user exit 53
- TBUILD command
 - handling errors 98
 - using 93
- tokens
 - CICSplex SM 44
 - user-defined 42
- TPARSE command
 - handling errors 98
 - using 93
- translating
 - command-level program 70
 - resource table attributes
 - in REXX program 94
 - RESPONSE and REASON values
 - using the command-level interface 80
 - using the run-time interface 80
- translation errors, REXX 97

U

- UNMARK command
 - using 29
- user tokens 42
- user-replaceable program
 - CICSplex SM API 69

V

- view
 - description 16

X
XICEREQ 69

Sending your comments to IBM

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

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

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

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

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

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

- By mail, to this address:
 - Information Development Department (MP095)
 - IBM United Kingdom Laboratories
 - Hursley Park
 - WINCHESTER,
 - Hampshire
 - SO21 2JN
 - United Kingdom
- By fax:
 - From outside the U.K., after your international access code use 44–1962–870229
 - From within the U.K., use 01962–870229
- Electronically, use the appropriate network ID:
 - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
 - IBMLink™: HURSLEY(IDRCF)
 - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

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



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC34-5457-00



Spine information:



CICS TS for OS/390

CICSplex® SM Application Programming Guide

Release 3