z/OS Communications Server

CMIP Services and Topology Agent Guide

Version 1 Release 7

SC31-8828-03

z/OS Communications Server

CMIP Services and Topology Agent Guide

Version 1 Release 7

SC31-8828-03

Note:
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 375.

Fourth Edition (September 2005)

This edition applies to Version 1 Release 7 of z/OS (5694-A01) and Version 1 Release 7 of z/OS.e (5655-G52) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You may send your comments to the following address.

International Business Machines Corporation
Attn: z/OS Communications Server Information Development
Department AKCA, Building 501
P.O. Box 12195, 3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195

You can send us comments electronically by using one of the following methods:

Fax (USA and Canada):

1+919-254-4028

Send the fax to "Attn: z/OS Communications Server Information Development"

Internet e-mail:

comsvrcf@us.ibm.com

World Wide Web:

http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number. Make sure to include the following in your comment or note:

- Title and order number of this document
- Page number or topic related to your comment

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

Contents

Figures

Tables

About this document

This document describes programming concepts and CMIP API functions that help application programmers write Common Management Information Protocol (CMIP) application programs that use VTAM® CMIP services. The information in this document supports both IPv6 and IPv4. Unless explicitly noted, information describes IPv4 networking protocol. IPv6 support is qualified within the text.

This document supports z/OS.e.

Who should read this document

Use this document if you are planning to write a manager or agent application program that uses VTAM CMIP services or the VTAM topology agent application program.

Before using this document, you should be familiar with the basic concepts of telecommunication, SNA, and VTAM. You should also be familiar with the following:

- C language programming
- Object-oriented terminology
- OSI network management

You should be familiar with the information in the *z/OS Communications Server: New Function Summary*. The *z/OS Communications Server: New Function Summary* contains an overview of CMIP services and the VTAM topology agent, including information about what these functions enable you to do and how to plan for these functions. This document gives you the new and changed user interfaces that enable you to use each function.

How this document is organized

This document contains the following parts and chapters:

- Part 1, "VTAM CMIP services," on page 1 provides reference information you need to write CMIP application programs. It contains the following chapters:
  - Chapter 1, "Introduction to Object Orientation and CMIP services," on page 3
  - Chapter 2, "Sample CMIP application program," on page 19
  - Chapter 3, "Overview of CMIP services API functions," on page 41
  - Chapter 4, "CMIP services API function syntax and operands," on page 53
  - Chapter 5, "Read queue exit routine," on page 87
  - Chapter 6, "Dequeue and release routines for data space storage," on page 91
  - Chapter 7, "Rules for constructing standard CMIP strings," on page 95
  - Chapter 8, "Examples of standard CMIP strings," on page 121
  - Chapter 9, "Create and delete requests," on page 129
  - Chapter 10, "VTAM-specific requests and responses," on page 133
  - Chapter 11, "Application-program-to-application-program security," on page 143
- Part 2, "VTAM topology agent," on page 147 explains what VTAM topology agent sends across the CMIP interface. It contains the following chapters:

- Chapter 12, “Introduction to VTAM topology agent,” on page 149
- Chapter 13, “OSI object classes and VTAM resources,” on page 151
- Chapter 14, “OSI operations,” on page 159
- Chapter 15, “VTAM topology monitoring,” on page 171
- Chapter 16, “Requesting specific resource data,” on page 219

- The appendixes provide information that you might find helpful. This document contains the following appendixes:
  - Appendix A, “C language header file (ACYAPHDH),” on page 229
  - Appendix B, “ASN.1 specification of the basic CMIP strings,” on page 239
  - Appendix C, “Error codes sent by CMIP services,” on page 263
  - Appendix D, “VTAM CMIP services compliance with related standards and profiles,” on page 299
  - Appendix E, “VTAM topology agent object and attribute tables,” on page 301
  - Appendix F, “VTAM topology agent attributes definition,” on page 313
  - Appendix G, “VTAMTOPO filtering option reporting,” on page 347
  - Appendix J, “Information APARs,” on page 369 lists information APARs for IP and SNA documents.
  - Appendix K, “Accessibility,” on page 373 describes accessibility features to help users with physical disabilities.
  - “Notices” on page 375 contains notices and trademarks used in this document.
  - “Bibliography” on page 387 contains descriptions of the documents in the z/OS® Communications Server library.

How to use this document

To use this document, you should be familiar with the basic concepts of telecommunications, SNA, and VTAM.

Determining whether a publication is current

As needed, IBM® updates its publications with new and changed information. For a given publication, updates to the hardcopy and associated BookManager® softcopy are usually available at the same time. Sometimes, however, the updates to hardcopy and softcopy are available at different times. The following information describes how to determine if you are looking at the most current copy of a publication:

- At the end of a publication’s order number there is a dash followed by two digits, often referred to as the dash level. A publication with a higher dash level is more current than one with a lower dash level. For example, in the publication order number GC28-1747-07, the dash level 07 means that the publication is more current than previous levels, such as 05 or 04.
- If a hardcopy publication and a softcopy publication have the same dash level, it is possible that the softcopy publication is more current than the hardcopy publication. Check the dates shown in the Summary of Changes. The softcopy publication might have a more recently dated Summary of Changes than the hardcopy publication.
- To compare softcopy publications, you can check the last two characters of the publication’s file name (also called the book name). The higher the number, the more recent the publication. Also, next to the publication titles in the CD-ROM booklet and the readme files, there is an asterisk (*) that indicates whether a publication is new or changed.

How to contact IBM service

For immediate assistance, visit this Web site:
http://www.software.ibm.com/network/commserver/support/

Most problems can be resolved at this Web site, where you can submit questions and problem reports electronically, as well as access a variety of diagnosis information.

For telephone assistance in problem diagnosis and resolution (in the United States or Puerto Rico), call the IBM Software Support Center anytime (1-800-IBM-SERV). You will receive a return call within 8 business hours (Monday – Friday, 8:00 a.m. – 5:00 p.m., local customer time).

Outside of the United States or Puerto Rico, contact your local IBM representative or your authorized IBM supplier.

If you would like to provide feedback on this publication, see "Communicating Your Comments to IBM" on page 405.

Conventions and terminology used in this document

For definitions of the terms and abbreviations used in this document, you can view the latest IBM terminology at the IBM Terminology Web site.

Clarification of notes

Information traditionally qualified as **Notes** is further qualified as follows:

Note Supplemental detail

Tip Offers shortcuts or alternative ways of performing an action; a hint

Guideline
Customary way to perform a procedure; stronger request than recommendation

Rule Something you must do; limitations on your actions

Restriction
Indicates certain conditions are not supported; limitations on a product or facility

Requirement
Dependencies, prerequisites

Result Indicates the outcome

Prerequisite and related information

z/OS Communications Server function is described in the z/OS Communications Server library. Descriptions of those documents are listed in "z/OS Communications Server information" on page 387, in the back of this document.

Required information

Before using this product, you should be familiar with TCP/IP, VTAM, MVS™, and UNIX® System Services.

Related information

This section contains subsections on:

Softcopy information

Softcopy publications are available in the following collections:

Titles	Order Number	Description
z/OS V1R7 Collection	SK3T-4269	This is the CD collection shipped with the z/OS product. It includes the libraries for z/OS V1R7, in both BookManager and PDF formats.
z/OS Software Products Collection	SK3T-4270	This CD includes, in both BookManager and PDF formats, the libraries of z/OS software products that run on z/OS but are not elements and features, as well as the *Getting Started with Parallel Sysplex®* bookshelf.
z/OS V1R7 and Software Products DVD Collection	SK3T-4271	This collection includes the libraries of z/OS (the element and feature libraries) and the libraries for z/OS software products in both BookManager and PDF format. This collection combines SK3T-4269 and SK3T-4270.
z/OS Licensed Product Library	SK3T-4307	This CD includes the licensed documents in both BookManager and PDF format.
System Center Publication IBM S/390® Redbooks™ Collection	SK2T-2177	This collection contains over 300 ITSO redbooks that apply to the S/390 platform and to host networking arranged into subject bookshelves.

Other documents

For information about z/OS products, refer to *z/OS Information Roadmap* (SA22-7500). The Roadmap describes what level of documents are supplied with each release of z/OS Communications Server, as well as describing each z/OS publication.

Relevant RFCs are listed in an appendix of the IP documents. Architectural specifications for the SNA protocol are listed in an appendix of the SNA documents.

The following table lists documents that might be helpful to readers.

Title	Number
DNS and BIND, Fourth Edition, O'Reilly and Associates, 2001	ISBN 0-596-00158-4
Routing in the Internet , Christian Huitema (Prentice Hall PTR, 1995)	ISBN 0-13-132192-7
sendmail, Bryan Costales and Eric Allman, O'Reilly and Associates, 2002	ISBN 1-56592-839-3
SNA Formats	GA27-3136
TCP/IP Illustrated, Volume I: The Protocols, W. Richard Stevens, Addison-Wesley Publishing, 1994	ISBN 0-201-63346-9

Title	Number
TCP/IP Illustrated, Volume II: The Implementation, Gary R. Wright and W. Richard Stevens, Addison-Wesley Publishing, 1995	ISBN 0-201-63354-X
TCP/IP Illustrated, Volume III, W. Richard Stevens, Addison-Wesley Publishing, 1995	ISBN 0-201-63495-3
TCP/IP Tutorial and Technical Overview	GG24-3376
Understanding LDAP	SG24-4986
z/OS Cryptographic Service System Secure Sockets Layer Programming	SC24-5901
z/OS Integrated Security Services Firewall Technologies	SC24-5922
z/OS Integrated Security Services LDAP Client Programming	SC24-5924
z/OS Integrated Security Services LDAP Server Administration and Use	SC24-5923
z/OS JES2 Initialization and Tuning Guide	SA22-7532
z/OS MVS Diagnosis: Procedures	GA22-7587
z/OS MVS Diagnosis: Reference	GA22-7588
z/OS MVS Diagnosis: Tools and Service Aids	GA22-7589
z/OS MVS Using the Subsystem Interface	SA22-7642
z/OS Program Directory	GI10-0670
z/OS UNIX System Services Command Reference	SA22-7802
z/OS UNIX System Services Planning	GA22-7800
z/OS UNIX System Services Programming: Assembler Callable Services Reference	SA22-7803
z/OS UNIX System Services User's Guide	SA22-7801
z/OS XL C/C++ Run-Time Library Reference	SA22-7821
zSeries OSA-Express Customer's Guide and Reference	SA22-7935

Redbooks

The following Redbooks might help you as you implement z/OS Communications Server.

Title	Number
Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration	SG24-5227
Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications	SG24-5228
Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 4: Connectivity and Routing	SG24-6516
Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security	SG24-6840
IBM Communication Controller Migration Guide	SG24-6298
IP Network Design Guide	SG24-2580
Managing OS/390® TCP/IP with SNMP	SG24-5866
Migrating Subarea Networks to an IP Infrastructure	SG24-5957
OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide: Volume 3: MVS Applications	SG24-5229
Secureway Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements	SG24–5631
SNA and TCP/IP Integration	SG24-5291
TCP/IP in a Sysplex	SG24-5235

Title	Number
TCP/IP Tutorial and Technical Overview	GG24-3376
Threadsafe Considerations for CICS	SG24-6351

Where to find related information on the Internet

z/OS

This site provides information about z/OS Communications Server release availability, migration information, downloads, and links to information about z/OS technology

http://www.ibm.com/servers/eserver/zseries/zos/

z/OS Internet Library

Use this site to view and download z/OS Communications Server documentation

http://www.ibm.com/servers/eserver/zseries/zos/bkserv/

IBM Communications Server product

The primary home page for information about z/OS Communications Server

http://www.software.ibm.com/network/commserver/

IBM Communications Server product support

Use this site to submit and track problems and search the z/OS Communications Server knowledge base for Technotes, FAQs, white papers, and other z/OS Communications Server information

http://www.software.ibm.com/network/commserver/support/

IBM Systems Center publications

Use this site to view and order Redbooks, Redpapers, and Technotes

http://www.redbooks.ibm.com/

IBM Systems Center flashes

Search the Technical Sales Library for Techdocs (including Flashes, presentations, Technotes, FAQs, white papers, Customer Support Plans, and Skills Transfer information)

http://www.ibm.com/support/techdocs/atsmastr.nsf

RFCs

Search for and view Request for Comments documents in this section of the Internet Engineering Task Force Web site, with links to the RFC repository and the IETF Working Groups Web page

http://www.ietf.org/rfc.html

Internet drafts

View Internet-Drafts, which are working documents of the Internet Engineering Task Force (IETF) and other groups, in this section of the Internet Engineering Task Force Web site

http://www.ietf.org/ID.html

Information about Web addresses can also be found in information APAR II11334.

DNS Web sites: For more information about DNS, see the following USENET news groups and mailing addresses:

USENET news groups
comp.protocols.dns.bind

BIND mailing lists
http://www.isc.org/ml-archives/

BIND Users

- Subscribe by sending mail to bind-users-request@isc.org.
- Submit questions or answers to this forum by sending mail to bind-users@isc.org.

BIND 9 Users (This list might not be maintained indefinitely.)

- Subscribe by sending mail to bind9-users-request@isc.org.
- Submit questions or answers to this forum by sending mail to bind9-users@isc.org.

Note: Any pointers in this publication to Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS elements and features, z/VM®, VSE/ESA™, and Clusters for AIX® and Linux™:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX System Services).
- Your Microsoft® Windows® workstation. You can install code to access IBM message explanations on the *z/OS Collection* (SK3T-4269), using LookAt from a Microsoft Windows command prompt (also known as the DOS command line).
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from a disk on your *z/OS Collection* (SK3T-4269), or from the LookAt Web site (click **Download**, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Using IBM Health Checker for z/OS

IBM Health Checker for z/OS is a z/OS component that installations can use to gather information about their system environment and system parameters to help identify potential configuration problems before they impact availability or cause outages. Individual products, z/OS components, or ISV software can provide checks that take advantage of the IBM Health Checker for z/OS framework. This book may refer to checks or messages associated with this component.

For additional information about checks and about IBM Health Checker for z/OS, see *IBM Health Checker for z/OS: User's Guide*. z/OS V1R4, V1R5, and V1R6 users can obtain the IBM Health Checker for z/OS from the z/OS Downloads page at http://www.ibm.com/servers/eserver/zseries/zos/downloads/.

SDSF also provides functions to simplify the management of checks. See *z/OS SDSF Operation and Customization* for additional information.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this document or any other z/OS Communications Server documentation:

- Go to the z/OS contact page at:

  http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

  There you will find the feedback page where you can enter and submit your comments.
- Send your comments by e-mail to comsvrcf@us.ibm.com. Be sure to include the name of the document, the part number of the document, the version of z/OS Communications Server, and, if applicable, the specific location of the text you are commenting on (for example, a section number, a page number or a table number).

Summary of changes

Summary of changes
for SC31-8828-03
z/OS Version 1 Release 7

This document contains information previously presented in SC31-8828–02, which supports z/OS Version 1 Release 5.

The information in this document includes descriptions of support for both IPv4 and IPv6 networking protocols. Unless explicitly noted, descriptions of IP protocol support concern IPv4. IPv6 support is qualified within the text.

New information

Support for model CDRSCs

- CDRSCs created from models are reported in topology but the models themselves are not.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

You may notice changes in the style and structure of some content in this document–for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

Summary of changes
for SC31-8828-02
z/OS Version 1 Release 5

This document contains information previously presented in SC31-8828–01, which supports z/OS Version 1 Release 2. The information in this document supports both IPv6 and IPv4. Unless explicitly noted, information describes IPv4 networking protocol. IPv6 support is qualified within the text.

New information

- luCollection data example for IPv6 address, see “luCollection (PU) snapshot example” on page 209.

Changed information

- connectionIDs for Enterprise Extender connections (both in port objects and logicalLink objects) include support for IPv6 addresses. See “connectionID” on page 322.
- tn3270ClientIpAddress syntax includes support for IPv6 addresses, see “tn3270ClientIpAddress” on page 343.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Starting with z/OS V1R5, you may notice changes in the style and structure of some content in this document–for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

Summary of changes
for SC31-8828-01
z/OS Version 1 Release 2

This document contains minor editorial changes.

Part 1. VTAM CMIP services

Chapter 1. Introduction to Object Orientation and CMIP services

VTAM Common Management Information Protocol (CMIP) services provides an open, standards-based access for network and systems management. Application programmers can use CMIP services to code manager and agent application programs to aid in systems management.

In pre-V4R3 releases of VTAM without CMIP, network application programs, such as the NetView® program, are frequently limited by two restrictions:

- They rely on the VTAMLST data set for information about the location of resources within the network.

  The VTAMLST data set gives an incomplete picture of the network because VTAMLST includes only resources that are pre-defined. It does not include APPN or subarea resources that are dynamically defined.
- They must reside with VTAM on the host.

  Because topology information cannot be gathered and sent to the NetView program at a remote location, the NetView program must reside with VTAM on the host.

With CMIP these two restrictions no longer apply for topology management. The VTAM topology agent is a part of VTAM that functions as a CMIP application program. Together with a manager application program, such as the NetView program, the topology agent provides data for the management of APPN and subarea topology. For a description of the VTAM topology agent, refer to Chapter 12, "Introduction to VTAM topology agent," on page 149. A *manager* application program is any CMIP application program that sends requests to other objects. An *agent* application program is any CMIP application program that processes requests from other objects.

You can write your own manager or agent application program by using the CMIP services application program interface (API). These application programs are not restricted to system management, VTAM, or SNA resources. For example, you can write an agent application program for the MVS system.

Object-Oriented view of resources

CMIP network management uses an object-oriented view of the resources in the network to simplify management.

This object-oriented system emphasizes the common properties of resources and reduces the requirement for a manager application program to understand all details of every type of resource in the network. Information about different network resources are represented by agent application programs in a common language, composed of CMIP strings. Manager application programs use this common language to communicate with agent application programs.

A user of a network management program issues commands to a manager application program, which sends CMIP requests to a network resource. Resources are represented by agent application programs, which accept the request and build

information about the network resource in the form of a CMIP response. The CMIP response is returned to the manager application program.

In VTAM CMIP services, managed network resources are called *objects*. An object is an *instance* of one or more *classes*. A CMIP class describes a type of resource in the network and specifies the properties that are common to instances of the class.

A CMIP class is described in GDMO templates. These templates are sets of declarations written in the GDMO language that describe one or more classes. The descriptions include properties of the objects in that class, such as:
- How the object is named
- What types of requests are valid for this object
- What attributes (characteristics) describe this object

Inheritance is the mechanism used in object-oriented systems to simplify interactions with objects by emphasizing common properties. A class can inherit characteristics or traits from one or more other classes. To inherit means to have all behaviors of another class. The class that inherits is a subclass of the class it inherits from. The class that is inherited from is the superclass of the class that inherits from it.

A subclass has all the behaviors of its superclass because it inherits from the superclass. In addition, a subclass has unique behaviors of its own.

Relationship between CMIP services and local application programs

Local application programs are CMIP application programs that reside with CMIP services on the host.

Local agent and manager CMIP application programs use character strings to represent requests and responses that flow between manager application programs and agent application programs.

The formats of CMIP requests and responses are described by syntaxes that are written in the ASN.1 language. The ASN.1 language describes data formats.

All requests and responses sent between CMIP services and local application programs are EBCDIC strings formatted according to string syntaxes written in the ASN.1 language, as shown in this simple syntax example:

```
StringA ::= SEQUENCE
            {
              level INTEGER,
              id    CHARACTER
            }
```

The syntax in the example is the rule for building a string of type StringA. Using that syntax and the ASN.1 standard, an application program can build a string of type StringA.

The following strings are examples of StringA strings:

```
(level 5, id 'A')
(level 1355, id 'Z')
(1244, M)
```

For more information on interpreting ASN.1 syntaxes, refer to Chapter 7, "Rules for constructing standard CMIP strings," on page 95.

Relationship between CMIP services and remote management systems

When CMIP requests and responses flow through the network, they are encoded in a hardware-independent format. CMIP services is available on machines with different word sizes (16-bit and 32-bit, for example) and different character string representations (ASCII and EBCDIC). This is hidden from application programs and from CMIP services because CMIP services encodes data from native format to a common format when it sends data across the network. It decodes data from the common format to native format when it receives data from the network.

Basic Encoding Rules (BER) is the common format that is used to encode CMIP information as it flows through the network. BER is not used between local agent application programs and manager application programs.

Overview of CMIP services

VTAM CMIP services is designed to provide information through VTAM to network and systems management application programs that conform to the OSI standards for systems management. CMIP services provides application writers a set of common functions that can be used to create CMIP agent and manager application programs more quickly than would otherwise be possible.

The relationship between CMIP agent and manager application programs is defined by the International Standards Organization (ISO) in terms of a managing system and a managed system. The managing system is the CMIP manager application program and the managed system is the CMIP agent application program.

With the functions provided by CMIP services, application programmers can write application programs that monitor resources in a network. Through CMIP services, a topology agent application program sends information about resources in the network to a topology manager application program that analyzes and displays the resources.

The VTAM topology agent, which resides on the VTAM host, is an agent application program that collects topology information to send to a manager application program through CMIP services. For information about the VTAM topology agent, refer to Chapter 12, "Introduction to VTAM topology agent," on page 149. Communication between the manager and agent application programs that are on different systems is over APPC sessions using Open System Interconnection (OSI) Common Management Information Protocol (CMIP) and Systems Network Architecture (SNA). For more information on CMIP over SNA, refer to *IBM SystemView® Mapping of OSI Upper Layers to MDS for CMIP over SNA for APPN and SNA Subarea Management*.

CMIP services enables communication between application programs by performing several functions for the application programs. The following sections describe these CMIP services tasks:

- Locates objects
- Registers objects
- Coordinates traffic
- Replicates scoped requests
- Filters events and routes them to manager application programs
- Provides security
- Creates and ends associations
- Manages associations

- Manages protocol data units (PDUs)
- Supports CMIP verbs and parameters

Locates objects

CMIP services allows your application program to target CMIP requests to local or remote objects without knowing where the objects reside, what their *application entity titles* are, or what their associations are. The directory resolves the object locations. Application programs can use the same code for local objects and for remote objects.

CMIP services maps an object instance, represented by its distinguished name, to the application entity title of the application entity that can be used to contact that object instance.

CMIP services performs the following tasks:

- Maps distinguished names to application entity titles by using a locally defined directory and either of the following methods:
  - Mappings (as defined by either the ACYDDF member of the SYS1.SISTCMIP data set or a CMIP algorithm) for distinguished names of specific formats to the application entity title that represents the distinguished name.
  - User-defined mappings for distinguished names of specific formats to the application entity title that represents the distinguished name and from application entity title to session address. See *z/OS Communications Server: SNA Network Implementation Guide* and *z/OS Communications Server: SNA Resource Definition Reference* for more information about user-defined mappings.
- Maps names to application entity titles by using a locally defined directory.

CMIP application programs can rely on CMIP services to provide this mapping. The application programs address the objects by their distinguished names only.

Only one mapping is allowed. You cannot define more than one application entity title for each distinguished name and cannot target more than one target system per application entity title.

Registers objects

CMIP services supports both manager and agent application programs. Any application program can act as both manager and agent. Each application program must have at least one object that it registers with CMIP services.

VTAM implements an instance of a system object defined by ISO/IEC 10165-2. The system object can be used by an application program to register subordinate objects if the name binding defined for the subordinate objects allows this. CMIP services provides the distinguished name of the local system object on return from the MIBConnect function (the CMIP services connection function) so that application programs can register subordinate objects to this system.

This distinguished name is especially useful if you are registering objects that are in the managerApplication class. Any application program choosing not to register under this system object can either register its own root object or can register under any currently registered object. CMIP services does not accept registration under non-existent managed objects. Instances can be registered under directory objects, which are created dynamically, or they can be registered under the root object.

The local system object is created when VTAM CMIP services is initialized and is therefore registered so long as VTAM CMIP services is active. As a result, this object provides a predictable, reliable anchor for creating and registering objects. It is highly recommended that event filter discriminator (EFD) objects be created under this system object. EFD objects are described in more detail under "Filters and routes events" and "Special considerations for topology manager application programs" on page 14.

CMIP services verifies proper names for object instances during object registration.

Because CMIP services is the only function that is aware of the tree structure for naming object instances, it processes scoped requests by replicating the incoming message for each object in the subtree specified by the scoping criteria. It does not filter messages.

When an object with multiple name bindings registers, CMIP services assigns it the first name binding it finds.

Coordinates traffic

CMIP services coordinates CMIP traffic within a local system. It includes an application program interface (API) and a management information base (MIB).

The MIB includes objects. CMIP services allows the local MIB to be used by several application programs. Each application program can implement one or more objects that comprise the MIB. The complete MIB is made up of all of the objects registered by the application programs. The CMIP application programs that use the MIB are called the agents or managers for the system.

Manager application programs do not have to understand where objects are located because VTAM directs the requests to the objects. Responses are matched with the requests and returned to the originator.

Replicates scoped requests

Requests that affect several application programs (or objects) within a particular scope are called **scoped requests**. Scoped requests are coordinated such that CMIP services provides the appropriate end responses when the affected objects have responded. CMIP services replicates scoped requests and directs them to the objects within each application program that fall within the scope of the request. Manager application programs on CMIP services can rely on CMIP services to find the base affected objects and deliver the request to the system containing that base object. At the receiving system, CMIP services delivers copies of the request to each affected object, coordinates the responses, and forwards the responses.

Filters and routes events

CMIP services filters **events** to forward them to any manager application programs that have indicated they want to see these events. The event reports contain information sent by a managed object relating to an event that has occurred within the managed object, such as a threshold violation or a change in configuration status.

Notifications are the conceptual messages that are sent by object instances to CMIP services. They do not have a destination initially. Notifications are specified using the notification syntax contained in Appendix B, "ASN.1 specification of the basic CMIP strings," on page 239. These messages are processed by CMIP services and if

there is an EFD object with a filter that matches that notification, they are converted into event reports that contain destinations.

In the case of inbound event reports destined for OSISMASE from CMIP services on products other than VTAM, CMIP services filters and routes event reports so that they can be forwarded to specific objects within the local system or to remote systems.

OSISMASE is the default application entity title for CMIP services. For information about OSISMASE, refer to *IBM SystemView Mapping of OSI Upper Layers to MDS for CMIP over SNA for APPN and SNA Subarea Management*. Inbound event reports targeted at application entities other than OSISMASE are routed directly to the object that registered the application entity.

VTAM CMIP services does not allow the creation of EFDs that reside in VTAM to specify OSISMASE as a destination. CMIP services on other products might allow OSISMASE as a destination.

Object instances do not have to be aware of destinations and filters for events because CMIP services does that.

CMIP services receives all notifications that are either sent by local object instances or received from other systems. CMIP services compares their attributes against matching criteria specified in each instance of the EFD managed object. For each EFD, if no match is found, the message is discarded. If a match is found, the destination specified in the event forwarding discriminator is attached to the message and it is processed further. The notification is converted to an unconfirmed event report. If eventTime was provided in the notification, it is copied to the event report, otherwise an eventTime is generated and included. The event report is sent to each destination in the destination list.

For a description of how a manager application program creates EFDs, refer to "Special considerations for topology manager application programs" on page 14.

CMIP services performs a set of functions common to all members of the EFD object class.

It also performs the functions defined in the IBM EFD subclass for allomorphic behavior of events. These functions are defined in *IBM SystemView Managed Resource Model Reference and Templates, Volume 1: Generic Definitions*. To support this additional behavior, each object instance that sends notifications must use the notification syntax to include with each notification the set of allomorphic superclasses that the object instance supports.

Confirmed event reports are not supported. When CMIP services receives a confirmed notification or a confirmed event report, CMIP services builds an ROER processing failure with no specific information.

EFD attributes that specify scheduling are ignored.

Objects can choose to register as individual application entities. If an application program registers as an application entity, then any event reports destined for that application entity are forwarded directly to that application program. Any event reports destined for the default application entity (OSISMASE) are routed to the local CMIP services. The creation of EFDs with a destination of OSISMASE is not valid and might be rejected by CMIP services.

To learn about registering application entity titles, refer to “Registering an application entity” on page 135.

Provides security

CMIP services provides two kinds of security. One kind of security is between association partners. It verifies that association partners have proper authorization to be in communication with each other. This kind of security defines which manager and agent application programs can communicate with each other. The system administrator controls this access by defining either only those partners that are allowed to request management functions or those that are to be specifically excluded. Wildcards and defaults can be used.

See *z/OS Communications Server: SNA Network Implementation Guide* and *z/OS Communications Server: SNA Resource Definition Reference* for more information about this type of security.

The other kind of security is across the API. The API security restricts access to application program that are not authorized to act as manager application programs or agent application programs. This security is implemented by a password, similar to the passwords used by traditional VTAM application programs.

For information about where the password is passed to the MIBConnect function, refer to “MIBConnect—MIB connection function” on page 56.

Creates and ends associations

An **association** is a logical connection between CMIP services on this host and CMIP services on another node or between CMIP services on this host and itself. An association between CMIP services on this host and itself is a local association. An association between CMIP services on this host and CMIP services on another node is a remote association.

Creating associations

Associations can be created in two ways:

- CMIP services can establish the association when it recognizes the need for one.
- An application program can establish an association with the ACF.Associate request, which is described under “Starting associations” on page 136.

Ending associations

An association can be ended by several methods:

- An application program can issue the ACF.Abort or ACF.Release request.
- CMIP services can end the association if it has been idle for 2 hours.
- The VTAM limited resources function (selective termination of idle LU 6.2 [APPC] sessions) sessions, can cause an association to be ended. For a description of the effect of selective termination on associations, refer to “Creating a dedicated association” on page 140.

Manages associations

CMIP services chooses the association across which to carry a particular message unless the application program overrides the default association by specifying an association on the MIBSendRequest function or the MIBSendCmipRequest function.

CMIP services chooses the association based on the type of message, the application context tied to the association, and the destination of the message. CMIP services enforces the application context against inbound messages.

It controls the minute-by-minute operations of associations by:

- Determining the type of the message and routing it to the correct element of CMIP services
- Maintaining the capabilities of the associations that exist
- Negotiating the capabilities of the associations
- Determining the correct association for a message
- Initiating an association for messages that are directed to object instances located on systems with which there are no associations
- Establishing a default association for messages that are directed to object instances on the local system
- Allowing local objects or application programs to monitor the state of associations
- Routing incoming messages to the correct function within CMIP services

CMIP services establishes associations. When establishing associations, it negotiates the application context to be used for that association. It ensures that the parameters are correct.

To ensure secure associations, VTAM CMIP services checks the directory definition file to see whether data-encryption-standard (DES)-based security or application-program-to-application-program security is in effect.

For an overview of the security function in VTAM CMIP services, refer to Chapter 11, "Application-program-to-application-program security," on page 143. See *z/OS Communications Server: SNA Network Implementation Guide* and *z/OS Communications Server: SNA Resource Definition Reference* for a description of the directory definition file.

Manages PDUs

As a service to local application programs, CMIP services determines whether protocol data units (PDUs) are properly formed and exchanged in the proper order. This service frees application programs from having to verify the PDUs themselves.

A PDU can have several types of errors. These include:

- A value is out of the legal range for the data type. The message is rejected.
- A tag is unrecognized in a SET™ value or SEQUENCE value. The message is rejected.

If a PDU suffers from several of these errors at one time, the most severe errors are processed first. When the message fails to be decoded, CMIP services tries to decode the Remote Operations Service Element (ROSE) header for the message. If the header can be decoded, the message is rejected.

In some cases, if the header cannot be decoded, the association is ended. This should not happen unless the message is totally destroyed.

CMIP services understands the messages that are exchanged with object instances. It maintains the list of outstanding requests that require replies and enforces that the CMIP strings it receives are correct.

CMIP services does not always ensure that duplicate linked-replies are not received.

Supports all CMIP verbs and most CMIP parameters

VTAM CMIP services supports the CMIP syntaxes as documented in Appendix B, "ASN.1 specification of the basic CMIP strings," on page 239 with certain exceptions. CMIP services supports all CMIP verbs:

- EVENT-REPORT
- GET
- SET
- ACTION
- CREATE
- DELETE
- CANCEL-GET

VTAM CMIP services does not support atomic synchronization. If atomic synchronization is specified, the CMIP request is responded to with a syncNotSupport error. VTAM CMIP services does not support the EFD scheduling attributes.

Requirements for application programs

As described in previous sections, VTAM CMIP services provides many services that free application programs from having to code many of the common CMIP functions. The application program is therefore allowed to focus on functions specific to the object instances it represents. The application program implements the behavior of its objects. It must:

- Code an APPL definition statement to define the application program to VTAM. See *z/OS Communications Server: SNA Resource Definition Reference* for information about the APPL definition statement.
- Connect to VTAM CMIP services using the MIBConnect function. When using the MIBConnect function, the application program must provide the address of its read queue exit routine. The read queue exit routine is required for application programs to communicate with CMIP services.

  It is highly recommended that you code a TPEND exit routine for VTAM to invoke when VTAM is terminating. If you code a TPEND exit routine, you must provide its address.
- Register at least one object instance using the MIBSendRegister function. An application program can register as many object instances as it represents. An object instance cannot be registered by more than one application program.
- Implement the behavior of the object instances it represents. CMIP services does not provide a repository for object attributes. Any CMIP operations targeting an object instance are delivered to the application program that registered that instance (or, in the case of a subtree manager, the application program that registered the subtree containing that instance).

  For example, a CMIP GET request is forwarded to the application program representing the objects targeted in the request. Those application programs are responsible for collecting the requested attributes, building them into the proper response, and sending them using the MIBSendCmipResponse function.

For scoped requests that affect object instances across multiple application programs, no coordination is needed between the application programs. CMIP services coordinates the requests for the application program. Your application program simply indicates that it has finished its part of the response by setting the last-in-chain attribute when invoking the MIBSendCmipResponse function. For hints on coding subtree managers refer to "Subtree managers." An application program can be both a manager and an agent, but it is helpful to separate them for the following discussion under "Types of application programs."

- Issue the MIBDisconnect function to disconnect the application from CMIP services.

Types of application programs

Different types of agent application programs have different rights and responsibilities. These types are defined by the capabilities that are requested when an object instance is registered. These types are:

- Basic application program, with no special capabilities
- Subtree manager application program
- Create handler application program

Basic application programs

A basic application program is one that represents one or more object instances, all of which are registered to CMIP services. The registering allows CMIP services to provide the most service because it can scope requests to each affected instance. A basic application program does not receive CMIP create requests to have new instances generated, but it can create and register any number of object instances. The trigger for creating these instances is the responsibility of the application program and is often dictated by the resources the application program must represent.

Subtree managers

A subtree manager is an application program that has assumed additional responsibilities. It supports any number of instances. It is not required to register any of them with CMIP services. It has requested and been granted ownership of a portion of the naming tree, which includes all instances contained within it.

All scoped indications that can include a member of the subtree owned by the subtree manager are passed to the subtree manager. It is responsible for managing scoping within its subtree and for creating all of the responses from its instances.

The subtree manager indicates to CMIP services that it has completed the responses from its supported instances. It cannot use the MIB variables &DN or &OC for any of its instances that are not registered. For information about MIB variables, refer to "MIB variable format" on page 98.

Once a subtree manager has registered itself, it establishes ownership of a subtree. At that point, no other application program can register objects within that subtree. Only the subtree manager can register additional objects within the subtree. For each leaf of the subtree that the subtree manager registers, it must first register all instances in that branch of the tree. An object cannot be registered unless its parent has been registered. Messages to an object within the subtree are assigned the local identifier of the subtree manager object. This is the local identifier that was explicitly requested.

A process that registers as a subtree manager can assume responsibility for one or more subtrees of the naming hierarchy. This capability allows the process to register only a small number of instances. A minimum of one instance is required. For each instance that it chooses not to register, the subtree manager must do the global-to-local name mapping and scoping functions that are provided by CMIP services for registered instances.

If many of the instances an application program represents are dynamic and changing frequently, it might be preferable for the application program to act as a subtree manager, instead of registering all of its instances. In that case, the overhead of registering the instances makes management too expensive to be practical.

Another example is when the application program chooses to control scoping itself. For example, if an agent application program is to receive scoped requests for a large number of objects, it might be better to receive a single scoped request. (A single scoped request is one that is not replicated by CMIP services.) A single scoped request might allow the request to be processed more efficiently internally.

Here we list one advantage and one disadvantage to registering as a subtree manager. The advantage is that a subtree manager can avoid registering some or all of its object instances and can control scoped operations. The disadvantage is that a subtree manager is required to assign names within the name space it owns. It must also ensure that the names are unique. It must perform all of the scoping function within its name space, a requirement that makes coding the application program more complicated.

When designing an application program, you must decide between writing additional code to provide these functions or registering all instances.

Create handlers

A create handler is an application program that assumes additional responsibilities. It registers to receive create messages for instances of a specific class. Registering allows create messages to be sent to a process that is capable of handling them. For information about the API function that registers a create handler, refer to “MIBSendRegister—MIB asynchronous registration function” on page 79.

Only one create handler can be registered per object class.

Special considerations for manager application programs

Manager application programs can have somewhat different requirements from agent application programs. A manager application program generally has no need to register any objects unless it needs to be the target of CMIP requests from other manager application programs. VTAM CMIP services requires that at least one object be registered. CMIP services does not require the object to be of a particular object class. The managerApplication object class has been defined for manager application programs that do not have a need for any specific class.

Manager application programs can base their management on the creation of EFDs so that they can receive CMIP event reports from managed systems. For a description of how to create the EFDs, refer to “Filters and routes events” on page 7. Such manager application programs must register to CMIP services as an application entity. The application entity title used must match the one specified in the destination list within the EFDs it creates on the managed systems. For

information about how an application can register as an application entity to CMIP services, refer to "Registering an application entity" on page 135.

Manager application programs that rely on CMIP event reports for monitoring objects at remote systems might need a mechanism to help them determine when the connection to the managed system is down. CMIP services gives application programs the ability to subscribe to associations. For example, a manager might want to subscribe to each association that was used for creating an EFD. The handle for each such association is returned in the response to the create request for the EFDs. For information on how an application program can subscribe to an association, refer to "Subscribing to association information" on page 133.

Special considerations for topology manager application programs

Usually, topology manager application programs need to know about specific resources or sets of resources, but do not want to receive event reports about all resources in a network. For CMIP services to know which resources the manager application program is interested in, the manager application program creates an EFD object and specifies a filter attribute for it to indicate which event reports are to be forwarded to the manager application program.

Therefore, to allow the VTAM topology agent to send only those notifications for resources that a topology manager application program is interested in, the following conditions must be met:

- VTAM must be started with the OSIEVENT=PATTERNS start option. See *z/OS Communications Server: SNA Resource Definition Reference* for a description of this start option.
- The manager application program must create EFD objects with filter attributes that follow the patterns that CMIP services recognizes. For a description of these patterns, refer to "Patterns of EFDs that CMIP services recognizes."

If the OSIEVENT=ALL start option is specified, the VTAM topology agent generates all possible notifications, as long as at least one EFD has been created. If no EFDs have been created, no notifications are generated.

If the OSIEVENT=NONE start option is specified, the VTAM topology agent generates no notifications.

Patterns of EFDs that CMIP services recognizes

If the filter attribute is specified according to the patterns described here and the OSIEVENT=PATTERNS start option is specified, CMIP services recognizes that the manager application program is interested in a particular resource or set of resources. CMIP services recognizes the following patterns:

- A filter specifies a certain object class but not a specific resource and the OSIEVENT=PATTERNS start option is specified.

  If the object class relates to VTAM topology, the VTAM topology agent forwards to CMIP services all notifications for all instances of that class. CMIP services then creates an event report and sends it to the manager application program if all criteria in the filter were met.
- A filter specifies a certain resource, with or without object class specified and the OSIEVENT=PATTERNS start option is specified.

If the object class relates to VTAM topology, the VTAM topology agent forwards notifications for that instance to CMIP services. CMIP services then creates an event report and sends it to the manager application program if all criteria in the filter were met.

- A filter is created locally by some manager application program to collect remote notifications using a filter similar to the one shown:

```
(item (equality (attributeId 1.3.18.0.0.1746, attributeValue
(mgr (distinguishedName '1.3.18.0.2.4.6=netid;2.9.3.2.7.4=(n
ame "cpname");1.3.18.0.0.2175=objectname')))))
```

  CMIP services assumes that such filters are not meant to collect topology information, so the presence of this EFD does not cause the topology agent to start generating notifications.

Specific object classes that CMIP services recognizes

Here are the object identifiers for the recognized classes:

1.3.18.0.0.1829
logicalUnit

1.3.18.0.0.2281
crossDomainResource

1.3.18.0.0.1803
luGroup

1.3.18.0.0.2267
definitionGroup

1.3.18.0.0.2085
logicalLink

1.3.18.0.0.2089
port

1.3.18.0.0.1844
t4Node

CMIP error handling

This section discusses the general VTAM CMIP error-handling scheme. It covers what types of errors can be detected and returned to invoking application programs and what types of general handling must occur when error conditions are returned.

The error handling scheme for the most part can be described in generic terms. Error handling specific to a given CMIP operation is described in the section that covers that operation.

General error handling

This section discusses how the Systems Management Application Entity (SMAE) portion of CMIP services handles remote operations CMIP (RO/CMIP) errors. In general, the error reporting mechanism is dictated by the area of CMIP services that detects the error.

Errors found during outbound CMIP processing

An outbound CMIP string is a CMIP string that is being sent from an application program to some destination.

In general, any error found in a request (confirmed and unconfirmed) or response in the originating SMAE is reported to the invoking application program by an asynchronous CMIP services API error code as a service error.

In the case where the destination of the CMIP string is on the same system as the origin of the CMIP string, some differences apply. If the CMIP string arrives at the presentation layer of CMIP services before an error is detected, the CMIP error is not reported as an API error code. In this case, once the CMIP string has passed the presentation layer and is back in the SMAE, the SMAE does not distinguish between same-system errors and different-system errors. The error in this case is handled as specified in the following list for inbound CMIP strings received from other systems. Refer to Chapter 3, "Overview of CMIP services API functions," on page 41 for a list of these API error codes.

The system that originated the outbound request can also receive errors detected on the destination system in the form of RO-REJECT(U), RO-REJECT(P), and RO-ERROR. These error types are passed to the application program if enough information is available for routing.

Errors found during inbound CMIP processing

An inbound CMIP string is a CMIP string (either request or response) that is being received from some CMIP sender. The sender can be on a different system or on the same system.

When the SMAE portion of CMIP services is the destination system of the CMIP request or response, error handling is handled as follows:

- If the error is found in ROSE, an RO-REJECT(P) is sent to the originating system.

  This is true for responses and requests (both confirmed and unconfirmed).
- If the error is found in CMISE, an RO-REJECT(U) is sent to the originating system.

  This is true for responses and requests (both confirmed and unconfirmed).
- For errors found in requests above CMISE in CMIP services, an RO-ERROR is returned if the request if confirmed.

  If the request is not confirmed, the request is discarded.
- For responses, the code above CMISE in CMIP services does not have any known error checking.

  If an error is found at this level, CMIP services attempts to pass the response to the appropriate object or discard the message if the message cannot be routed.
- If an application program detects an error during CMIP request processing, an RO-ERROR is returned if the request if confirmed.

  If the request is not confirmed, the request is discarded.

  For confirmed requests, the actual errors returned are to be defined by the application program, such as the VTAM topology agent. Refer to "Responding to CMIP requests" on page 161 for more information on how the VTAM topology agent handles such errors.
- If an application program detects an error during CMIP response processing, the error handling processing is defined by the application program. Refer to "Responding to CMIP requests" on page 161 for more information on how the VTAM topology agent handles such errors.

CMIP sequencing for separate CMIP operations

CMIP flows that relate to separate CMIP operations could flow between the agent application program and the manager application program in any order. The VTAM topology agent and CMIP services do not attempt to ensure that such CMIP strings, generated as the result of separate operations, are sequenced and delivered based on order of events or processing. For example, a notification that is generated by VTAM after a GET response is built could actually be received by the manager application program before the GET response.

Therefore, the manager application program should not rely on order of receipt as an indication of order of processing at the agent application program. There is no correlation between order of processing by the agent application program and time of receipt by the manager application program.

Chapter 2. Sample CMIP application program

Many of the aspects of writing a CMIP application program can be explained using a sample application program. This chapter presents a CMIP application program that sends a simple CMIP request to another application program on any host.

The purpose of this application program is to determine whether or not CMIP services is active on a specific host in the network. In other words, this is a ping application program for CMIP over SNA much as APING is a ping application program for APPC. The sample program implements this by sending a CMIP GET request to the system object on that host. The system object should always exist, either as part of CMIP services or as part of a CMIP application program. If an error occurs bringing up an association to the remote CMIP services, then either the specified host is unreachable or CMIP services is not active on that host. Otherwise, the specified host is reachable and CMIP services is active. Errors returned by the remote system object itself are unimportant.

Note: The system object is implemented by VTAM as part of CMIP services, so it is always present if VTAM CMIP services is active.

The sample application program is comprised of the following source files:

ACYCMS1C
This C language module is the main logic of the application program. It calls several different API functions to communicate with CMIP services.

ACYCMS2A
This assembler language module is the read queue exit routine for the application program.

ACYCMS3A
This assembler language module is used to obtain the address of an API function in LPALIB.

ACYCMS4A
This assembler language module is used to switch the application program task into supervisor state.

ACYCMS5A
This assembler language module is used to wait on an ECB.

ACYCMS6A
This assembler language module is the TPEND exit routine for the CMIPPING application program.

ACYCMS7A
This assembler language module is used to switch the application program task into problem state.

"ACYCMS1C source file" on page 22 is the main logic for the CMIPPING application program.

Note: To facilitate reading on any host terminal and printing on any host printer, trigraph sequences have been used for square brackets. These sequences are ??(for left square bracket and ??) for right square bracket.

An outline of processing in function main is listed here:

1. Make sure that the user has provided the required parameters to the program.
   a. TargetNetid is the SNA netID of the host that will be pinged.
   b. TargetNauname is the SNA NAU name (in this case, a CP name or SSCP name) of the host that will be pinged.
   c. ApplName is the ACB name used by CMIPPING when issuing MIBConnect.
   d. Password, if provided, is the ACB password as specified on the APPL statement.
2. Load the addresses of the API functions which are used.

   This program uses MIBConnect, MIBDisconnect, MIBSendCmipRequest, and MIBSendRegister. ACYCMS3A is used to find the addresses of all of the API functions.
3. Switch to supervisor state.

   The caller of API routines must be in supervisor state. ACYCMS4A is responsible for issuing the MODESET system macro to switch the task to supervisor state.
4. Connect to CMIP services.

   A CMIP Application must issue MIBConnect before calling any other MIB API functions. MIBConnect opens an ACB on behalf of the calling application program, initializes the connection with CMIP services, and returns various information to the caller.
5. Register a managerApplication object.

   Even though CMIPPING does not need to represent the behavior of any objects for the purposes of the application program, it must register an object nonetheless because CMIP services requires that requests be issued by an object that it knows about. The managerApplication object class was defined for manager application programs that use the registered object only as the source of requests.

   Before calling the MIBSendRegister function, it first builds the name of the managerApplication object. The name of the system object on this system, returned by MIBConnect, is used to build the name of this object.
6. Wait for the registration message from CMIP services.

   ACYCMS6A is called to wait on an ECB. This ECB will be posted by the read queue exit routine (ACYCMS2A) when it is called by CMIP services. The next message to arrive should be the registration response.
7. Parse the registration response message from CMIP services.

   If the msg_type field in the APIhdr is API_REG_ACCEPT and the invokeId field in the APIhdr is the one returned by MIBSendRegister, then the registration succeeded.
8. Send a GET request to the system object on the target host.

   This first builds the name of the remote system object to which a GET request will be sent. It then builds the entire CMIP string representing the GET argument.

   The CMIP string is passed to MIBSendCmipRequest, which will send the GET request to CMIP services for processing.
9. Wait for the GET response message from CMIP services.

   ACYCMS6A is called again to wait until the read queue exit routine posts an ECB to wake up the main task. The next message should be the GET response.

10. Parse the GET response to determine whether or not CMIP services is active on the target host.

    If the invokeId field in the APIhdr is the one returned by MIBSendCmipRequest and the msg_type field in the APIhdr is API_MSG, then the request was received by a remote CMIP services. Determining whether or not the system object was available on the remote system and was able to processing the request would require parsing the string portion of the response. That is beyond the scope of this application program.
11. Disconnect from CMIP services. If MIBConnect succeeded, MIBDisconnect should be called — even if some other error happened in between.
12. Exit the application program.

"ACYCMS2A source file" on page 29 is the read queue exit routine for the CMIPPING application program. An outline of processing in the exit follows:

1. The VTAM reason code is always stored in the user data control block so that the main task of CMIPPING (ACYCMS1C) can find out why the read queue exit routine was driven.
2. If the reason code is zero, meaning that VTAM passed data to the read queue exit routine, that data will be copied to the buffer in the user data control block.
3. If the reason code is something other than zero, a message will be generated. The likely scenario is that CMIP services is terminating.
4. The read queue exit routine posts an ECB which is being waited on by the main task of CMIPPING in order to wake it up.
5. The read queue exit routine returns zero to VTAM, telling VTAM that it was able to successfully process the message.

Note: In a real CMIP application program read queue exit routine, you probably need additional buffer space to pass messages to the main task. Some CMIP requests can result in many messages being returned by CMIP services, one after another. It is unlikely that an application program designed like CMIPPING would see all of the messages, since they would arrive more quickly than the main task could be dispatched and process each one.

"ACYCMS3A source file" on page 31 is a utility module to load the addresses of the API functions on behalf of the CMIPPING application program. The only processing to perform is to load the address of each routine into a control block passed by the caller (ACYCMS1C).

Note: This module does not check return codes from the LOAD macro and always returns zero. This is not appropriate for a real application program, since those modules may not be installed in LPALIB.

"ACYCMS4A source file" on page 34 is a utility module to switch into supervisor state. The only processing to perform is to invoke the MODESET assembler macroinstruction.

Note: CMIPPING must be authorized for the MODESET to work.

"ACYCMS5A source file" on page 35 is a utility module to wait on a specified ECB. The only processing to perform is to invoke the WAIT assembler macroinstruction.

"ACYCMS6A source file" on page 36 is the TPEND exit routine for the CMIPPING application program. All it does is display the reason code passed by VTAM.

"ACYCMS7A source file" on page 38 is a utility module to switch into problem state. The only processing to perform is to invoke the MODESET assembler macroinstruction.

ACYCMS1C source file

```
/*************************************************************/
/*                                                           */
/* MEMBER NAME: ACYCMS1C                                     */
/*                                                           */
/* DESCRIPTIVE NAME: Sample CMIP Application                 */
/*                                                           */
//*                                                          */
/* COPYRIGHT: LICENSED MATERIALS - PROPERTY OF IBM           */
/*                                                           */
/*            "RESTRICTED MATERIALS OF IBM"                  */
/*                                                           */
/*            5695-117 (C) COPYRIGHT IBM CORP. 1994          */
/*                                                           */
/* MEMBER CATEGORY:  Sample CMIP application                 */
/*                                                           */
/*                                                           */
/*************************************************************/

/*
 * CMIPPING - Sample C language CMIP application
 *
 * This sample application can be used to determine if CMIP Services
 * is active on a specified host in the network.
 *
 * It illustrates some of the concepts involved in writing a CMIP
 * application for use with VTAM.
 *
 * Notes: To facilitate reading on any host terminal and printing on
 *        any host printer, trigraph sequences have been used for
 *        square brackets.  These sequences are "??(" for left square
 *        bracket and "??)" for right square bracket.
 */

#pragma csect(code, "ACYCMS1C")
#pragma csect(static,"SCYCMS1C")

/*********************************************************************/
/* C Standard Library include files                                  */
/*********************************************************************/

#include #include #include
/*********************************************************************/
/* VTAM include files                                                */
/*********************************************************************/

#include "acyaphdh.h"              /* VTAM MIB API interface        */

/*********************************************************************/
/* type declarations                                                 */
/*********************************************************************/

/*********************************************************************/
/* An instance of MIBAddresses_t is passed to ACYCMS3A, which fills  */
/* it in with actual addresses of the MIB API functions, which are   */
/* loaded from LPALIB.                                               */
/*********************************************************************/
```

```
typedef struct MIBAddresses_tag
{
  MIBConnect_t                *MIBConnect;
  MIBDisconnect_t             *MIBDisconnect;
  MIBSendRegister_t           *MIBSendRegister;
  MIBSendDeleteRegistration_t *MIBSendDeleteRegistration;
  MIBSendRequest_t            *MIBSendRequest;
  MIBSendResponse_t           *MIBSendResponse;
  MIBSendCmipRequest_t        *MIBSendCmipRequest;
  MIBSendCmipResponse_t       *MIBSendCmipResponse;
} MIBAddresses_t;

/*********************************************************************/
/* The address of an instance of ReadQueueExitData_t is passed to    */
/* MIBConnect.  CMIP Services then passes that same address to the   */
/* Read Queue Exit each time it is called.  That allows sharing of   */
/* data between the Read Queue Exit and this main task.              */
/*********************************************************************/

typedef struct ReadQueueExitData_tag
{
  int ECB;
  int ReasonCode;
  char Buffer ??(16384??);
} ReadQueueExitData_t;

/*********************************************************************/
/* external functions                                                */
/*********************************************************************/

/*********************************************************************/
/* ACYCMS2A is the Read Queue Exit for this sample CMIP application. */
/* Only the address of this routine is needed in C.                  */
/*********************************************************************/

extern void ACYCMS2A(void);

/*********************************************************************/
/* ACYCMS3A finds the addresses of routines in LPALIB via the LOAD   */
/* assembler macroinstruction.                                       */
/*********************************************************************/

extern int ACYCMS3A(MIBAddresses_t *);

/*********************************************************************/
/* ACYCMS4A switches to supervisor state via the MODESET assembler   */
/* macroinstruction.                                                 */
/*********************************************************************/

extern void ACYCMS4A(void);

/*********************************************************************/
/* ACYCMS5A waits on an ECB via the WAIT assembler macroinstruction. */
/* It will be passed the address of the same ECB which the Read      */
/* Queue Exit will post so that the Read Queue Exit can "wake up"    */
/* this task when data is available.                                 */
/*********************************************************************/

extern void ACYCMS5A(int *ECB);

/*********************************************************************/
/* ACYCMS6A is the TPEND exit for this application.                  */
/*********************************************************************/

extern void ACYCMS6A(void);
```

```
/*********************************************************************/
/* ACYCMS7A switches to problem state via the MODESET assembler      */
/* macroinstruction.                                                 */
/*********************************************************************/

extern void ACYCMS7A(void);

/*********************************************************************/
/* constants                                                         */
/*********************************************************************/

#define APPL_NAME "CMIPPING"        /* name of this application as
                                       used in messages              */

/*********************************************************************/
/* data types                                                        */
/*********************************************************************/

typedef void *LocalId_t;            /* Local identifiers, associated
                                       with registered objects, must
                                       have a size between 1 and 8
                                       bytes.  CMIPPING uses four-
                                       byte local identifiers of type
                                       void *.                       */

/*********************************************************************/
/* Input parameters:                                                 */
/*   (1)  Netid of target CMIP Services                              */
/*   (2)  Nauname of target CMIP Services                            */
/*   (3)  Application name to use (valid ACB name)                   */
/*   (4)  Optional password                                          */
/*********************************************************************/

int main(int argc,const char **argv)
{
  APIhdr *APIhdr_ptr;
  char CMIP_StringArgument ??(512??);
  char MyObjectName        ??(120??);
  char RemoteSystemObject  ??(120??);
  char SMAE                ??(100??);
  char SystemObject        ??(100??);
  const char *ApplName;
  const char *Password;
  const char *TargetNauname;
  const char *TargetNetid;
  char *VTAM_Release;
  int Connected, rc;
  int LinkId;
  MIBAddresses_t APIs;
  unsigned int InvokeId;
  ReadQueueExitData_t ReadQueueExitData;
  size_t SMAE_Size, SystemObjectSize;
  unsigned int ACB_Info;
  LocalId_t *MyObjectId;

  rc = 0;

  memset(&ReadQueueExitData,0,sizeof(ReadQueueExitData));

  if (argc != 4 && argc != 5)
  {
    fprintf(stderr,
            "Usage: " APPL_NAME " Netid Nauname Applname \n"
            "\n"
            "       " APPL_NAME " is used to determine whether or not\n"
            "       there is an active CMIP Services on a SNA host\n"
            "       specified by Netid and Nauname.\n"
```

```
          "\n"
          "       Applname is the ACB name used by this program.\n"
          "       Netid and Nauname specify the target SNA host.\n"
          "       Password (optional) is the ACB password.\n"
          "\n"
          "       " APPL_NAME " cannot continue.\n");
  rc = 1;
}

if (rc == 0)                      /* If no errors so far...        */
{
  TargetNetid   = argv ??(1??);   /* first parm passed to program  */
  TargetNauname = argv ??(2??);   /* second parm                   */
  ApplName      = argv ??(3??);   /* third parm                    */

  if (argc == 5)                  /* If a password was provided... */
    Password    = argv ??(4??);   /* fourth parm                   */
  else
    Password    = NULL;

  ACYCMS4A();                     /* You must be in supervisor
                                     state to call VTAM MIB API
                                     routines.                     */

  SMAE_Size = sizeof(SMAE);
  SystemObjectSize = sizeof(SystemObject);
  MyObjectId = (void *)"Anything you want"; /* The local identifier
                                     for the object registered by
                                     this application  is the
                                     address of this string.       */
}

/*********************************************************************/
/* Obtain addresses of API routines used by this program.            */
/*********************************************************************/

if (rc == 0)                      /* If no errors so far...        */
{
  rc = ACYCMS3A(&APIs);

  if (rc != 0)
  {
    fprintf(stderr,
            "The address of an API routine could not be loaded\n"
            "from LPALIB.\n"
            "\n"
            APPL_NAME " cannot continue.\n");
  }
}

if (rc == 0)                      /* If no errors so far...        */
{
  rc = APIs.MIBConnect(0,         /* always zero for this release  */
                       &LinkId,   /* MIBConnect will fill in LinkId
                                     with a handle to the
                                     connection.                   */
                       65536,     /* maximum number of outstanding
                                     requests                      */
                       ApplName,  /* ACB name                      */
                       (void *)ACYCMS6A, /* TPEND exit             */
                       (void *)ACYCMS2A, /* address of the Read
                                     Queue Exit                    */
                       &SMAE_Size, /* On input, this is the size of
                                     the SMAE buffer.  On output,
                                     this is the length of the SMAE
                                     name.                         */
                       SMAE,      /* This is where MIBConnect will
```

```
                                    store the SMAE name (if there
                                    is enough room).             */
                          &SystemObjectSize, /* On input, this is the
                                    size of the System Object name
                                    buffer.  On output, this is the
                                    length of the System Object
                                    name.                        */
                          SystemObject, /* This is where MIBConnect will
                                    store the System Object name
                                    (if there is enough room).   */
                          (int)&ReadQueueExitData, /* This will be
                                    provided to this appl's Read
                                    Queue Exit by CMIP Services.  */
                          &ACB_Info, /* If an error occurs opening the
                                    ACB, this will contain the
                                    OPEN ACB error code.         */
                          &VTAM_Release, /* MIBConnect will store the
                                    address of the VTAM release
                                    level here.                  */
                          Password,  /* ACB password              */
                          0,         /* dataspace not used        */
                          NULL,      /* dataspace not used        */
                          sizeof(LocalId_t), /* size of local ids
                                    for all objects registered by
                                    this application             */
                          0);        /* no special options specified  */

  Connected = rc == 0;            /* Remember whether or not
                                     MIBConnect was successful.   */

  if (rc != 0)
  {
    fprintf(stderr,
            "MIBConnect returned %d.\n"
            "\n"
            APPL_NAME " cannot continue.\n",
            rc);
  }
}

if (rc == 0)                      /* If no errors so far...       */
{
  /*****************************************************************/
  /* Build the distinguished name of the object that will be       */
  /* registered.  It is named directly "under" the System Object,  */
  /* so its name is the system object name concatenated with one   */
  /* more RelativeDistinguishedName.                               */
  /*                                                               */
  /* A short form distinguished name (DN) will be built.  Note     */
  /* that CMIP Services can handle distinguished names from        */
  /* applications in either short or long form.  Applications can  */
  /* elect to receive distinguished names from CMIP Services in    */
  /* short form by specifying SHORT_NAMES as the last parameter to */
  /* MIBConnect.  By default, applications receive distinguished   */
  /* names in long form.                                           */
  /*****************************************************************/

  strcpy(MyObjectName,SystemObject);
  strcat(MyObjectName,";1.3.18.0.0.2175=");
  strcat(MyObjectName,ApplName);

  puts("Here is the object being registered:");
  puts(MyObjectName);

  rc = APIs.MIBSendRegister(LinkId, /* This is the handle returned by
                                    MIBConnect.                  */
                            &InvokeId, /* MIBSendRegister will store
```

```
                                        an invoke id, or correlator, for
                                        the registration request here.*/
                                &MyObjectId, /* This is the address of
                                        the local id to be associated
                                        with this object.            */
                                "1.3.18.0.0.2155", /* This is the object
                                        class of this object.        */
                                DN_OF_INSTANCE, /* This parameter must
                                        have this value.             */
                                MyObjectName, /* This is the dist.
                                        name of this object.         */
                                NULL, /* Use default name binding.    */
                                0,    /* no special capabilities      */
                                0,    /* no allomorphs                */
                                NULL, /* no allomorphs                */
                                0,    /* not a create handler for any
                                        class                        */
                                NULL); /* not a create handler        */
  if (rc != 0)
  {
    fprintf(stderr,
            "MIBSendRegister returned %d.\n"
            "\n"
            APPL_NAME " cannot continue.\n",
            rc);
  }
}

if (rc == 0)                        /* If o.k. so far...            */
{
  ReadQueueExitData.ECB = 0;
  ACYCMS5A(&ReadQueueExitData.ECB);
  if (ReadQueueExitData.ReasonCode == 0) /* If data was received...*/
    APIhdr_ptr = (APIhdr *)ReadQueueExitData.Buffer;
  else
    rc = ReadQueueExitData.ReasonCode;
}

/*********************************************************************/
/* Parse the response from registration to see if it was o.k.        */
/*********************************************************************/

if (rc == 0)                        /* If o.k. so far...            */
{
  if ((APIhdr_ptr->msg_type != API_REG_ACCEPT) ||
      (APIhdr_ptr->invokeId != InvokeId)) /* If an error
                                        occurred...                  */
  {
    rc = 1;
    fprintf(stderr,
            "An unexpected response was received from object\n"
            "registration.\n"
            "\n"
            APPL_NAME " cannot continue.\n");
  }
}

if (rc == 0)                        /* If o.k. so far...            */
{
  /*******************************************************************/
  /* Build CMIP GET request string here using the netid and          */
  /* nauname of the target host.                                     */
  /*******************************************************************/

  strcpy(RemoteSystemObject,"1.3.18.0.2.4.6=");
  strcat(RemoteSystemObject,TargetNetid);
  strcat(RemoteSystemObject,";2.9.3.2.7.4=(name ");
```

```
  strcat(RemoteSystemObject,TargetNauname);
  strcat(RemoteSystemObject,")");

  strcpy(CMIP_StringArgument,
         "("
           "baseManagedObjectClass 2.9.3.2.3.13, "
           "baseManagedObjectInstance "
            "(distinguishedName '");
  strcat(CMIP_StringArgument,RemoteSystemObject);
  strcat(CMIP_StringArgument,"'),"
           "attributeIdList "
            "(2.9.3.2.7.35,2.9.3.2.7.5)"
         ")");

  rc = APIs.MIBSendCmipRequest(LinkId, /* handle returned by
                                       MIBConnect                    */
                               3, /* operation value is GET         */
                               CMIP_StringArgument,
                               &MyObjectId,
                               NULL,
                               DS_NOT_PROVIDED,
                               NULL,
                               &InvokeId);
  if (rc != 0)
  {
    fprintf(stderr,
            "MIBSendCmipRequest returned %d.\n"
            "\n"
            APPL_NAME " cannot continue.\n",
            rc);
  }
}

if (rc == 0)                      /* If o.k. so far...              */
{
  ReadQueueExitData.ECB = 0;
  ACYCMS5A(&ReadQueueExitData.ECB);
  if (ReadQueueExitData.ReasonCode == 0) /* If data was received...*/
    APIhdr_ptr = (APIhdr *)ReadQueueExitData.Buffer;
  else
    rc = ReadQueueExitData.ReasonCode;
}

if (rc == 0)
{
  /*****************************************************************/
  /* Display whether or not the GET was successful.                */
  /*****************************************************************/

  if ((APIhdr_ptr->msg_type == API_MSG) &&
      (APIhdr_ptr->invokeId == InvokeId))
  {
    /***************************************************************/
    /* Technically, the message can be a CMIP error message which  */
    /* could state that the system object doesn't exist or that    */
    /* the system object can't handle the request.  Since this     */
    /* program only checks to see if the specified CMIP Services   */
    /* can be contacted, a CMIP error message will not be          */
    /* considered a problem.                                       */
    /***************************************************************/

    puts("The remote CMIP Services was contacted successfully.");
  }
  else
  {
    fprintf(stderr,
            "The remote CMIP Services could not be contacted.\n");
```

```
      rc = 1;
    }
  }

  if (Connected)
  {
    rc = APIs.MIBDisconnect(LinkId, /* This is the handle returned by
                                       MIBDisconnect.               */
                       &ACB_Info);  /* If an error occurs closing the
                                       ACB, MIBDisconnect will store
                                       the CLOSE ACB error code here.*/
    if (rc != 0)
    {
      fprintf(stderr,
              "MIBDisconnect returned %d.\n",
              rc);
    }
  }

  ACYCMS7A();                       /* You must be in problem
                                       state to exit cleanly.       */

  fprintf(stderr,
          APPL_NAME " is exiting with return code %d.\n",
          rc);

  return rc;
}
```

ACYCMS2A source file

```
*/************************************************************/
*/*                                                          */
*/* MEMBER NAME: ACYCMS2A                                    */
*/*                                                          */
*/* DESCRIPTIVE NAME: Read Queue Exit for sample CMIP        */
*/*                   application                            */
*/*                                                          */
*/*                                                          */
*/* COPYRIGHT: LICENSED MATERIALS - PROPERTY OF IBM          */
*/*                                                          */
*/*            "RESTRICTED MATERIALS OF IBM"                 */
*/*                                                          */
*/*            5695-117 (C) COPYRIGHT IBM CORP. 1994         */
*/*                                                          */
*/* MEMBER CATEGORY:  Sample CMIP application                */
*/*                                                          */
*/************************************************************/
         TITLE ' /*****************************************************00001000
               ********'                                                00002000
ACYCMS2A CSECT ,                                             0001 00003000
ACYCMS2A AMODE 24                                            0001 00004000
ACYCMS2A RMODE 24                                            0001 00005000
@MAINENT DS    0H                                            0001 00006000
         USING *,@15                                         0001 00007000
         B     @PROLOG                                       0001 00008000
         DC    AL1(16)                                       0001 00009000
         DC    C'ACYCMS2A  95.125'                           0001 00010000
         DROP  @15                                                00011000
@PROLOG  STM   @14,@12,12(@13)                               0001 00012000
         LR    @12,@15                                       0001 00013000
@PSTART  EQU   ACYCMS2A                                      0001 00014000
         USING @PSTART,@12                                   0001 00015000
         ST    @13,@SA00001+4                                0001 00016000
         LA    @14,@SA00001                                  0001 00017000
         ST    @14,8(,@13)                                   0001 00018000
         LR    @13,@14                                       0001 00019000
```

```
*                                                                    0014 00020000
*/*****************************************************************/ 00021000
*/*                      MAINLINE                                   */ 00022000
*/*****************************************************************/ 00023000
*                                                                    0014 00024000
*R10 = R1;                          /* Saves Pointer to Parameters   */ 00025000
*                                                                    0014 00026000
         LR    R10,R1                                                0014 00027000
*GLB_DATA.GLB_ReasonCode = VTAM_REASON; /* Tell the main task why the   00028000
*                                     Read Queue Exit was driven.   */ 00029000
*                                                                    0015 00030000
         L     @04,VTAM_REASON(,R10)                                 0015 00031000
         ST    @04,GLB_REASONCODE(,R06_USER_DATA)                    0015 00032000
*IF (VTAM_REASON = 0) THEN          /* If data is available to be  0016 00033000
*                                     copied...                     */ 00034000
         LTR   @04,@04                                               0016 00035000
         BNZ   @RF00016                                              0016 00036000
*  DO;                                                               0017 00037000
*    R11 = VTAM_LENGTH;                                              0018 00038000
         L     @05,VTAM_LENGTH(,R10)                                 0018 00039000
         LR    R11,@05                                               0018 00040000
*    R3 = VTAM_LENGTH;                                               0019 00041000
         LR    R3,@05                                                0019 00042000
*    R2 = ADDR(GLB_Buffer);                                          0020 00043000
         LA    R2,GLB_BUFFER(,R06_USER_DATA)                         0020 00044000
*    TMP_R10 = R10;                                                  0021 00045000
         LR    @07_TMP_R10,R10                                       0021 00046000
*    R10 = VTAM_APIHDR_PTR;                                          0022 00047000
         L     R10,VTAM_APIHDR_PTR(,R10)                             0022 00048000
*    MVCL(R2,R10);                                                   0023 00049000
         MVCL  R2,R10                                                0023 00050000
*    R10 = TMP_R10;                                                  0024 00051000
         LR    R10,@07_TMP_R10                                       0024 00052000
*  END;                                                              0025 00053000
*ELSE                                                               0026 00054000
*  DO;                                                               0026 00055000
         B     @RC00016                                              0026 00056000
@RF00016 DS    0H                                                    0027 00057000
*    GEN (WTO 'CMIPPING: Read Queue Exit driven with reason <> 0!');    00058000
@GS00027 DS    0H                                                    0027 00059000
         WTO 'CMIPPING: Read Queue Exit driven with reason <> 0!'       00060000
@GE00027 DS    0H                                                    0028 00061000
*  END;                                                              0028 00062000
*                                                                    0028 00063000
*/*****************************************************************/     00064000
*/* Wake up the main task by posting the ECB which it is         */     00065000
*/* waiting on.                                                  */     00066000
*/*****************************************************************/     00067000
*                                                                    0029 00068000
*R1 = ADDR(GLB_ECB);                                                0029 00069000
@RC00016 LR    R1,R06_USER_DATA                                      0029 00070000
*GEN;                                                               0030 00071000
*                                                                    0030 00072000
@GS00030 DS    0H                                                    0030 00073000
  POST (1),X'FFFF'                                                      00074000
@GE00030 DS    0H                                                    0031 00075000
*RETURN CODE(0);                    /* Return to VTAM.               */ 00076000
*                                                                    0031 00077000
         SLR   @15,@15                                               0031 00078000
         L     @13,4(,@13)                                           0031 00079000
         L     @14,12(,@13)                                          0031 00080000
         LM    @00,@12,20(@13)                                       0031 00081000
         BR    @14                                                   0031 00082000
*END ACYCMS2A;                                                      0032 00083000
@DATA    DS    0H                                                         00084000
         DS    0F                                                         00085000
@SA00001 DS    18F                                                        00086000
```

```
         DS    0F                                                       00087000
         LTORG                                                          00088000
         DS    0D                                                       00089000
@DYNSIZE EQU   0                                                        00090000
@00      EQU   0                                                        00091000
@01      EQU   1                                                        00092000
@02      EQU   2                                                        00093000
@03      EQU   3                                                        00094000
@04      EQU   4                                                        00095000
@05      EQU   5                                                        00096000
@06      EQU   6                                                        00097000
@07      EQU   7                                                        00098000
@08      EQU   8                                                        00099000
@09      EQU   9                                                        00100000
@10      EQU   10                                                       00101000
@11      EQU   11                                                       00102000
@12      EQU   12                                                       00103000
@13      EQU   13                                                       00104000
@14      EQU   14                                                       00105000
@15      EQU   15                                                       00106000
@07_TMP_R10 EQU @07                                                     00107000
R0       EQU   @00                                                      00108000
R1       EQU   @01                                                      00109000
R2       EQU   @02                                                      00110000
R3       EQU   @03                                                      00111000
R06_USER_DATA EQU @06                                                   00112000
R10      EQU   @10                                                      00113000
R11      EQU   @11                                                      00114000
VTAM_PARM_LIST EQU 0,20,C'C'                                            00115000
VTAM_REASON EQU VTAM_PARM_LIST,4,C'F'                                   00116000
VTAM_APIHDR_PTR EQU VTAM_PARM_LIST+4,4,C'A'                             00117000
VTAM_LENGTH EQU VTAM_PARM_LIST+12,4,C'F'                                00118000
VTAM_APIHDR EQU 0,,C'C'                                                 00119000
GLB_DATA EQU   0,16392,C'C'                                             00120000
GLB_ECB  EQU   GLB_DATA,4,C'F'                                          00121000
GLB_REASONCODE EQU GLB_DATA+4,4,C'F'                                    00122000
GLB_BUFFER EQU GLB_DATA+8,16384,C'C'                                    00123000
         AGO   .UNREF                                                   00124000
VTAM_MSG_TYPE EQU VTAM_PARM_LIST+16,4,C'F'                              00125000
VTAM_STR_HEADER_PTR EQU VTAM_PARM_LIST+8,4,C'A'                         00126000
.UNREF   ANOP                                                           00127000
         DS    0D                                                       00128000
@ENDDATA EQU   *                                                        00129000
@MODLEN  EQU   @ENDDATA-ACYCMS2A                                        00130000
         END   ,(PL/X-370,0103,95125)                                   00131000
```

ACYCMS3A source file

```
*/************************************************************/
*/*                                                          */
*/* MEMBER NAME: ACYCMS3A                                    */
*/*                                                          */
*/* DESCRIPTIVE NAME: Load addresses of MIB API functions    */
*/*                   for sample CMIP application            */
*/*                                                          */
*/*                                                          */
*/* COPYRIGHT: LICENSED MATERIALS - PROPERTY OF IBM          */
*/*                                                          */
*/*            "RESTRICTED MATERIALS OF IBM"                 */
*/*                                                          */
*/*            5695-117 (C) COPYRIGHT IBM CORP. 1994         */
*/*                                                          */
*/* MEMBER CATEGORY:  Sample CMIP application                */
*/*                                                          */
*/************************************************************/
         TITLE ' /*****************************************************00001000
               ********'                                                00002000
```

```
ACYCMS3A CSECT ,                                                        0001 00003000
ACYCMS3A AMODE 24                                                       0001 00004000
ACYCMS3A RMODE 24                                                       0001 00005000
@MAINENT DS    0H                                                       0001 00006000
         USING *,@15                                                    0001 00007000
         B     @PROLOG                                                  0001 00008000
         DC    AL1(16)                                                  0001 00009000
         DC    C'ACYCMS3A  95.125'                                      0001 00010000
         DROP  @15                                                          00011000
@PROLOG  STM   @14,@12,12(@13)                                          0001 00012000
         LR    @12,@15                                                  0001 00013000
@PSTART  EQU   ACYCMS3A                                                 0001 00014000
         USING @PSTART,@12                                              0001 00015000
*                                                                       0007 00016000
*/*****************************************************************/ 00017000
*/*                                                                 */ 00018000
*/*                          MAINLINE                               */ 00019000
*/*                                                                 */ 00020000
*/*****************************************************************/ 00021000
*                                                                       0007 00022000
*/*****************************************************************/ 00023000
*/* MVS will abnormally terminate the task if a routine cannot be    */ 00024000
*/* found.  "Good" code would use the ERRET parameter on the LOAD    */ 00025000
*/* macro to provide an exit to be called if the specified module    */ 00026000
*/* cannot be found.  Using that capability, this routine could      */ 00027000
*/* return a bad return code instead of having the task terminated   */ 00028000
*/* when a routine can't be found.                                   */ 00029000
*/*****************************************************************/ 00030000
*                                                                       0007 00031000
*R10 = R1;                          /* Free up R1 since LOAD will  0007 00032000
*                                      clobber it.                  */ 00033000
*                                                                       0007 00034000
         LR    R10,R1                                                   0007 00035000
*RFY R1 UNRSTD;                                                         0008 00036000
*                                                                       0008 00037000
*GEN CODE SETS(R0,R1) DEFS(ACYAPCNP) (LOAD EP=ACYAPCNP);                0009 00038000
@GS00009 DS    0H                                                       0009 00039000
         LOAD EP=ACYAPCNP                                                   00040000
@GE00009 DS    0H                                                       0010 00041000
*MIBConnect = R0;                                                       0010 00042000
*                                                                       0010 00043000
         L     @11,PARM_PTR(,R10)                                       0010 00044000
         ST    R0,MIBCONNECT(,@11)                                      0010 00045000
*GEN CODE SETS(R0,R1) DEFS(ACYAPD1P) (LOAD EP=ACYAPD1P);                0011 00046000
@GS00011 DS    0H                                                       0011 00047000
         LOAD EP=ACYAPD1P                                                   00048000
@GE00011 DS    0H                                                       0012 00049000
*MIBDisconnect = R0;                                                    0012 00050000
*                                                                       0012 00051000
         L     @11,PARM_PTR(,R10)                                       0012 00052000
         ST    R0,MIBDISCONNECT(,@11)                                   0012 00053000
*GEN CODE SETS(R0,R1) DEFS(ACYAPRGP) (LOAD EP=ACYAPRGP);                0013 00054000
@GS00013 DS    0H                                                       0013 00055000
         LOAD EP=ACYAPRGP                                                   00056000
@GE00013 DS    0H                                                       0014 00057000
*MIBSendRegister = R0;                                                  0014 00058000
*                                                                       0014 00059000
         L     @11,PARM_PTR(,R10)                                       0014 00060000
         ST    R0,MIBSENDREGISTER(,@11)                                 0014 00061000
*GEN CODE SETS(R0,R1) DEFS(ACYAPDRP) (LOAD EP=ACYAPDRP);                0015 00062000
@GS00015 DS    0H                                                       0015 00063000
         LOAD EP=ACYAPDRP                                                   00064000
@GE00015 DS    0H                                                       0016 00065000
*MIBSendDeleteRegistration = R0;                                        0016 00066000
*                                                                       0016 00067000
         L     @11,PARM_PTR(,R10)                                       0016 00068000
         ST    R0,MIBSENDDELETEREGISTRATION(,@11)                       0016 00069000
```

```
*GEN CODE SETS(R0,R1) DEFS(ACYAPQRP) (LOAD EP=ACYAPQRP);          0017 00070000
@GS00017 DS    0H                                                 0017 00071000
         LOAD EP=ACYAPQRP                                              00072000
@GE00017 DS    0H                                                 0018 00073000
*MIBSendRequest = R0;                                             0018 00074000
*                                                                 0018 00075000
         L     @11,PARM_PTR(,R10)                                 0018 00076000
         ST    R0,MIBSENDREQUEST(,@11)                            0018 00077000
*GEN CODE SETS(R0,R1) DEFS(ACYAPRSP) (LOAD EP=ACYAPRSP);          0019 00078000
@GS00019 DS    0H                                                 0019 00079000
         LOAD EP=ACYAPRSP                                              00080000
@GE00019 DS    0H                                                 0020 00081000
*MIBSendResponse = R0;                                            0020 00082000
*                                                                 0020 00083000
         L     @11,PARM_PTR(,R10)                                 0020 00084000
         ST    R0,MIBSENDRESPONSE(,@11)                           0020 00085000
*GEN CODE SETS(R0,R1) DEFS(ACYAPQCP) (LOAD EP=ACYAPQCP);          0021 00086000
@GS00021 DS    0H                                                 0021 00087000
         LOAD EP=ACYAPQCP                                              00088000
@GE00021 DS    0H                                                 0022 00089000
*MIBSendCmipRequest = R0;                                         0022 00090000
*                                                                 0022 00091000
         L     @11,PARM_PTR(,R10)                                 0022 00092000
         ST    R0,MIBSENDCMIPREQUEST(,@11)                        0022 00093000
*GEN CODE SETS(R0,R1) DEFS(ACYAPCPP) (LOAD EP=ACYAPCPP);          0023 00094000
@GS00023 DS    0H                                                 0023 00095000
         LOAD EP=ACYAPCPP                                              00096000
@GE00023 DS    0H                                                 0024 00097000
*MIBSendCmipResponse = R0;                                        0024 00098000
*                                                                 0024 00099000
         L     @11,PARM_PTR(,R10)                                 0024 00100000
         ST    R0,MIBSENDCMIPRESPONSE(,@11)                       0024 00101000
*RETURN CODE(0);                    /* Assume that no error            00102000
*                                      occurred.                   */ 00103000
*                                                                 0025 00104000
         SLR   @15,@15                                            0025 00105000
         L     @14,12(,@13)                                       0025 00106000
         LM    @00,@12,20(@13)                                    0025 00107000
         BR    @14                                                0025 00108000
*END ACYCMS3A;                                                    0026 00109000
@DATA    DS    0H                                                      00110000
         DS    0F                                                      00111000
         DS    0F                                                      00112000
         LTORG                                                         00113000
         DS    0D                                                      00114000
@DYNSIZE EQU   0                                                      00115000
@00      EQU   0                                                      00116000
@01      EQU   1                                                      00117000
@02      EQU   2                                                      00118000
@03      EQU   3                                                      00119000
@04      EQU   4                                                      00120000
@05      EQU   5                                                      00121000
@06      EQU   6                                                      00122000
@07      EQU   7                                                      00123000
@08      EQU   8                                                      00124000
@09      EQU   9                                                      00125000
@10      EQU   10                                                     00126000
@11      EQU   11                                                     00127000
@12      EQU   12                                                     00128000
@13      EQU   13                                                     00129000
@14      EQU   14                                                     00130000
@15      EQU   15                                                     00131000
R0       EQU   @00                                                    00132000
R1       EQU   @01                                                    00133000
R10      EQU   @10                                                    00134000
MIBADDRESSES_T EQU 0,32,C'C'                                          00135000
MIBCONNECT EQU MIBADDRESSES_T,4,C'A'                                  00136000
```

```
MIBDISCONNECT EQU MIBADDRESSES_T+4,4,C'A'                                00137000
MIBSENDREGISTER EQU MIBADDRESSES_T+8,4,C'A'                              00138000
MIBSENDDELETEREGISTRATION EQU MIBADDRESSES_T+12,4,C'A'                   00139000
MIBSENDREQUEST EQU MIBADDRESSES_T+16,4,C'A'                              00140000
MIBSENDRESPONSE EQU MIBADDRESSES_T+20,4,C'A'                             00141000
MIBSENDCMIPREQUEST EQU MIBADDRESSES_T+24,4,C'A'                          00142000
MIBSENDCMIPRESPONSE EQU MIBADDRESSES_T+28,4,C'A'                         00143000
PARM_PTR EQU   0,4,C'A'                                                  00144000
         DS    0D                                                        00145000
@ENDDATA EQU   *                                                        00146000
@MODLEN  EQU   @ENDDATA-ACYCMS3A                                        00147000
         END   ,(PL/X-370,0103,95125)                                    00148000
```

ACYCMS4A source file

```
*/************************************************************/
*/*                                                          */
*/* MEMBER NAME: ACYCMS4A                                    */
*/*                                                          */
*/* DESCRIPTIVE NAME: Switch to supervisor state for sample  */
*/*                   CMIP application                       */
*/*                                                          */
*/*                                                          */
*/* COPYRIGHT: LICENSED MATERIALS - PROPERTY OF IBM          */
*/*                                                          */
*/*            "RESTRICTED MATERIALS OF IBM"                 */
*/*                                                          */
*/*            5695-117 (C) COPYRIGHT IBM CORP. 1994         */
*/*                                                          */
*/* MEMBER CATEGORY:  Sample CMIP application                */
*/*                                                          */
*/************************************************************/
         TITLE ' /*************************************************00001000
               ********'                                           00002000
ACYCMS4A CSECT ,                                          0001 00003000
ACYCMS4A AMODE 24                                         0001 00004000
ACYCMS4A RMODE 24                                         0001 00005000
@MAINENT DS    0H                                         0001 00006000
         USING *,@15                                       0001 00007000
         B     @PROLOG                                     0001 00008000
         DC    AL1(16)                                     0001 00009000
         DC    C'ACYCMS4A  95.125'                         0001 00010000
         DROP  @15                                              00011000
@PROLOG  STM   @14,@12,12(@13)                             0001 00012000
         LR    @12,@15                                     0001 00013000
@PSTART  EQU   ACYCMS4A                                    0001 00014000
         USING @PSTART,@12                                 0001 00015000
*                                                          0002 00016000
@GS00002 DS    0H                                          0002 00017000
         MODESET MODE=SUP                                       00018000
@GE00002 DS    0H                                          0003 00019000
*END ACYCMS4A;                                             0003 00020000
@EL00001 DS    0H                                          0003 00021000
@EF00001 DS    0H                                          0003 00022000
@ER00001 LM    @14,@12,12(@13)                             0003 00023000
         BR    @14                                         0003 00024000
@DATA    DS    0H                                               00025000
         DS    0F                                               00026000
         DS    0F                                               00027000
         LTORG                                                  00028000
         DS    0D                                               00029000
@DYNSIZE EQU   0                                                00030000
@00      EQU   0                                                00031000
@01      EQU   1                                                00032000
@02      EQU   2                                                00033000
@03      EQU   3                                                00034000
@04      EQU   4                                                00035000
```

```
@05      EQU   5                                                    00036000
@06      EQU   6                                                    00037000
@07      EQU   7                                                    00038000
@08      EQU   8                                                    00039000
@09      EQU   9                                                    00040000
@10      EQU   10                                                   00041000
@11      EQU   11                                                   00042000
@12      EQU   12                                                   00043000
@13      EQU   13                                                   00044000
@14      EQU   14                                                   00045000
@15      EQU   15                                                   00046000
         DS    0D                                                   00047000
@ENDDATA EQU   *                                                  00048000
@MODLEN  EQU   @ENDDATA-ACYCMS4A                                   00049000
         END   ,(PL/X-370,0103,95125)                              00050000
```

ACYCMS5A source file

```
*/************************************************************/
*/*                                                          */
*/* MEMBER NAME: ACYCMS5A                                    */
*/*                                                          */
*/* DESCRIPTIVE NAME: WAIT on ECB for sample CMIP application */
*/*                                                          */
*/*                                                          */
*/* COPYRIGHT: LICENSED MATERIALS - PROPERTY OF IBM          */
*/*                                                          */
*/*            "RESTRICTED MATERIALS OF IBM"                 */
*/*                                                          */
*/*            5695-117 (C) COPYRIGHT IBM CORP. 1994         */
*/*                                                          */
*/* MEMBER CATEGORY:  Sample CMIP application                */
*/*                                                          */
*/************************************************************/
         TITLE ' /*********************************************************00001000
               ********'                                                  00002000
ACYCMS5A CSECT ,                                              0001 00003000
ACYCMS5A AMODE 24                                             0001 00004000
ACYCMS5A RMODE 24                                             0001 00005000
@MAINENT DS    0H                                             0001 00006000
         USING *,@15                                          0001 00007000
         B     @PROLOG                                        0001 00008000
         DC    AL1(16)                                        0001 00009000
         DC    C'ACYCMS5A  95.125'                            0001 00010000
         DROP  @15                                                 00011000
@PROLOG  STM   @14,@12,12(@13)                                0001 00012000
         LR    @12,@15                                        0001 00013000
@PSTART  EQU   ACYCMS5A                                       0001 00014000
         USING @PSTART,@12                                    0001 00015000
*                                                             0009 00016000
*/*****************************************************************/ 00017000
*/*                                                               */ 00018000
*/*                       MAINLINE                                */ 00019000
*/*                                                               */ 00020000
*/*****************************************************************/ 00021000
*                                                             0009 00022000
*R2 = R1;                                                     0009 00023000
*                                                             0009 00024000
         LR    R2,R1                                          0009 00025000
*R1 = ADDR(THE_ECB);                                          0010 00026000
*                                                             0010 00027000
         L     R1,THE_ECB_PTR(,R2)                            0010 00028000
*GEN EXIT;                                                    0011 00029000
*                                                             0011 00030000
@GS00011 DS    0H                                             0011 00031000
  WAIT 1,ECB=(1)                                                   00032000
@GE00011 DS    0H                                             0012 00033000
```

```
*RETURN CODE(0);                                                0012 00034000
*                                                               0012 00035000
         SLR   @15,@15                                          0012 00036000
         L     @14,12(,@13)                                     0012 00037000
         LM    @00,@12,20(@13)                                  0012 00038000
         BR    @14                                              0012 00039000
*END ACYCMS5A;                                                  0013 00040000
@DATA    DS    0H                                                    00041000
         DS    0F                                                    00042000
         DS    0F                                                    00043000
         LTORG                                                       00044000
         DS    0D                                                    00045000
@DYNSIZE EQU   0                                                     00046000
@00      EQU   0                                                     00047000
@01      EQU   1                                                     00048000
@02      EQU   2                                                     00049000
@03      EQU   3                                                     00050000
@04      EQU   4                                                     00051000
@05      EQU   5                                                     00052000
@06      EQU   6                                                     00053000
@07      EQU   7                                                     00054000
@08      EQU   8                                                     00055000
@09      EQU   9                                                     00056000
@10      EQU   10                                                    00057000
@11      EQU   11                                                    00058000
@12      EQU   12                                                    00059000
@13      EQU   13                                                    00060000
@14      EQU   14                                                    00061000
@15      EQU   15                                                    00062000
R0       EQU   @00                                                   00063000
R1       EQU   @01                                                   00064000
R2       EQU   @02                                                   00065000
R14      EQU   @14                                                   00066000
R15      EQU   @15                                                   00067000
THE_ECB  EQU   0,4,C'F'                                              00068000
THE_ECB_PTR EQU 0,4,C'A'                                             00069000
         DS    0D                                                    00070000
@ENDDATA EQU   *                                                     00071000
@MODLEN  EQU   @ENDDATA-ACYCMS5A                                     00072000
         END   ,(PL/X-370,0103,95125)                                00073000
```

ACYCMS6A source file

```
*/************************************************************/
*/*                                                          */
*/* MEMBER NAME: ACYCMS6A                                    */
*/*                                                          */
*/* DESCRIPTIVE NAME: TPEND exit for sample CMIP application  */
*/*                                                          */
*/*                                                          */
*/* COPYRIGHT: LICENSED MATERIALS - PROPERTY OF IBM          */
*/*                                                          */
*/*            "RESTRICTED MATERIALS OF IBM"                 */
*/*                                                          */
*/*            5695-117 (C) COPYRIGHT IBM CORP. 1994         */
*/*                                                          */
*/* MEMBER CATEGORY:  Sample CMIP application                */
*/*                                                          */
*/************************************************************/
         TITLE ' /*********************************************************00001000
               ********'                                                  00002000
ACYCMS6A CSECT ,                                                0001 00003000
ACYCMS6A AMODE 24                                               0001 00004000
ACYCMS6A RMODE 24                                               0001 00005000
@MAINENT DS    0H                                               0001 00006000
         USING *,@15                                            0001 00007000
         B     @PROLOG                                          0001 00008000
```

```
         DC    AL1(16)                                                0001 00009000
         DC    C'ACYCMS6A  95.125'                                    0001 00010000
         DROP  @15                                                         00011000
@PROLOG  LR    @12,@15                                                0001 00012000
@PSTART  EQU   ACYCMS6A                                               0001 00013000
         USING @PSTART,@12                                            0001 00014000
*                                                                     0005 00015000
*                                                                     0005 00016000
*/*****************************************************************/ 00017000
*/*                                                                 */ 00018000
*/*                      MAINLINE                                   */ 00019000
*/*                                                                 */ 00020000
*/*****************************************************************/ 00021000
*                                                                     0005 00022000
*SELECT (REASON_CODE);               /* Issue message based on reason   00023000
*                                       code                          */ 00024000
         L     @02,REASON_CODE(,R1)                                   0005 00025000
         LTR   @02,@02                                                0005 00026000
         BM    @RT00014                                               0005 00027000
         BE    @RT00006                                               0005 00028000
         LA    @00,12                                                 0005 00029000
         CR    @02,@00                                                0005 00030000
         BH    @RT00014                                               0005 00031000
         BE    @RT00012                                               0005 00032000
         IC    @02,@CB00064(@02)                                      0005 00033000
         SLL   @02,2                                                  0005 00034000
         B     @GL00001(@02)                                          0005 00035000
@GL00001 B     @RT00014                                               0005 00036000
         B     @RT00008                                               0005 00037000
         B     @RT00010                                               0005 00038000
*WHEN (0)                                                            0006 00039000
@RT00006 DS    0H                                                     0007 00040000
*  GEN;                                                              0007 00041000
*                                                                    0007 00042000
@GS00007 DS    0H                                                     0007 00043000
   WTO '                              '                                   00044000
   WTO 'CMIPPING TPEND DRIVEN:  REASON CODE=00'                           00045000
   WTO '                              '                                   00046000
@GE00007 DS    0H                                                     0008 00047000
*WHEN (4)                                                            0008 00048000
         B     @RC00005                                               0008 00049000
@RT00008 DS    0H                                                     0009 00050000
*  GEN;                                                              0009 00051000
*                                                                    0009 00052000
@GS00009 DS    0H                                                     0009 00053000
   WTO '                              '                                   00054000
   WTO 'CMIPPING TPEND DRIVEN:  REASON CODE=04'                           00055000
   WTO '                              '                                   00056000
@GE00009 DS    0H                                                     0010 00057000
*WHEN (8)                                                            0010 00058000
         B     @RC00005                                               0010 00059000
@RT00010 DS    0H                                                     0011 00060000
*  GEN;                                                              0011 00061000
*                                                                    0011 00062000
@GS00011 DS    0H                                                     0011 00063000
   WTO '                              '                                   00064000
   WTO 'CMIPPING TPEND DRIVEN:  REASON CODE=08'                           00065000
   WTO '                              '                                   00066000
@GE00011 DS    0H                                                     0012 00067000
*WHEN (12)                                                           0012 00068000
         B     @RC00005                                               0012 00069000
@RT00012 DS    0H                                                     0013 00070000
*  GEN;                                                              0013 00071000
*                                                                    0013 00072000
@GS00013 DS    0H                                                     0013 00073000
   WTO '                              '                                   00074000
   WTO 'CMIPPING TPEND DRIVEN:  REASON CODE=12'                           00075000
```

```
   WTO '                                '                                00076000
@GE00013 DS    0H                                                  0014 00077000
*OTHERWISE                                                         0014 00078000
         B     @RC00005                                            0014 00079000
@RT00014 DS    0H                                                  0015 00080000
*  GEN;                                                            0015 00081000
*                                                                  0015 00082000
@GS00015 DS    0H                                                  0015 00083000
   WTO '                                '                             00084000
   WTO 'CMIPPING TPEND DRIVEN:  UNEXPECTED REASON CODE!'             00085000
   WTO '                                '                             00086000
@GE00015 DS    0H                                                  0016 00087000
*END;                                                              0016 00088000
*                                                                  0016 00089000
@RC00005 DS    0H                                                  0017 00090000
*RETURN CODE(RC);                    /* return to VTAM               */ 00091000
*                                                                  0017 00092000
@EL00001 DS    0H                                                  0017 00093000
@EF00001 DS    0H                                                  0017 00094000
@ER00001 BR    @14                                                 0017 00095000
*END ACYCMS6A;                                                     0018 00096000
@DATA    DS    0H                                                       00097000
         DS    0F                                                       00098000
         DS    0F                                                       00099000
         LTORG                                                          00100000
         DS    0D                                                       00101000
@CB00064 DC    XL12'000000000100000002000000'                           00102000
@DYNSIZE EQU   0                                                        00103000
@00      EQU   0                                                        00104000
@01      EQU   1                                                        00105000
@02      EQU   2                                                        00106000
@03      EQU   3                                                        00107000
@04      EQU   4                                                        00108000
@05      EQU   5                                                        00109000
@06      EQU   6                                                        00110000
@07      EQU   7                                                        00111000
@08      EQU   8                                                        00112000
@09      EQU   9                                                        00113000
@10      EQU   10                                                       00114000
@11      EQU   11                                                       00115000
@12      EQU   12                                                       00116000
@13      EQU   13                                                       00117000
@14      EQU   14                                                       00118000
@15      EQU   15                                                       00119000
R1       EQU   @01                                                      00120000
RC       EQU   @15                                                      00121000
TPEND_PARM_LIST EQU 0,8,C'C'                                           00122000
REASON_CODE EQU TPEND_PARM_LIST+4,4,C'F'                               00123000
         AGO   .UNREF                                                  00124000
ACB_PTR  EQU   TPEND_PARM_LIST,4,C'A'                                  00125000
.UNREF   ANOP                                                          00126000
         DS    0D                                                       00127000
@ENDDATA EQU   *                                                       00128000
@MODLEN  EQU   @ENDDATA-ACYCMS6A                                       00129000
         END   ,(PL/X-370,0103,95125)                                  00130000
```

ACYCMS7A source file

```
*/************************************************************/
*/*                                                          */
*/* MEMBER NAME: ACYCMS7A                                    */
*/*                                                          */
*/* DESCRIPTIVE NAME: Switch to problem state for sample CMIP */
*/*                   application                            */
*/*                                                          */
*/*                                                          */
*/* COPYRIGHT: LICENSED MATERIALS - PROPERTY OF IBM          */
```

```
*/*                                                                 */
*/*           "RESTRICTED MATERIALS OF IBM"                         */
*/*                                                                 */
*/*           5695-117 (C) COPYRIGHT IBM CORP. 1994                 */
*/*                                                                 */
*/* MEMBER CATEGORY:  Sample CMIP application                       */
*/*                                                                 */
*/*****************************************************************/
         TITLE ' /*****************************************************00001000
               ********'                                                00002000
ACYCMS7A CSECT ,                                              0001 00003000
ACYCMS7A AMODE 24                                             0001 00004000
ACYCMS7A RMODE 24                                             0001 00005000
@MAINENT DS    0H                                             0001 00006000
         USING *,@15                                          0001 00007000
         B     @PROLOG                                        0001 00008000
         DC    AL1(16)                                        0001 00009000
         DC    C'ACYCMS7A  95.125'                            0001 00010000
         DROP  @15                                                 00011000
@PROLOG  STM   @14,@12,12(@13)                                0001 00012000
         LR    @12,@15                                        0001 00013000
@PSTART  EQU   ACYCMS7A                                       0001 00014000
         USING @PSTART,@12                                    0001 00015000
*                                                             0002 00016000
@GS00002 DS    0H                                             0002 00017000
         MODESET MODE=PROB                                         00018000
@GE00002 DS    0H                                             0003 00019000
*END ACYCMS7A;                                                0003 00020000
@EL00001 DS    0H                                             0003 00021000
@EF00001 DS    0H                                             0003 00022000
@ER00001 LM    @14,@12,12(@13)                                0003 00023000
         BR    @14                                            0003 00024000
@DATA    DS    0H                                                   00025000
         DS    0F                                                   00026000
         DS    0F                                                   00027000
         LTORG                                                      00028000
         DS    0D                                                   00029000
@DYNSIZE EQU   0                                                    00030000
@00      EQU   0                                                    00031000
@01      EQU   1                                                    00032000
@02      EQU   2                                                    00033000
@03      EQU   3                                                    00034000
@04      EQU   4                                                    00035000
@05      EQU   5                                                    00036000
@06      EQU   6                                                    00037000
@07      EQU   7                                                    00038000
@08      EQU   8                                                    00039000
@09      EQU   9                                                    00040000
@10      EQU   10                                                   00041000
@11      EQU   11                                                   00042000
@12      EQU   12                                                   00043000
@13      EQU   13                                                   00044000
@14      EQU   14                                                   00045000
@15      EQU   15                                                   00046000
         DS    0D                                                   00047000
@ENDDATA EQU   *                                                    00048000
@MODLEN  EQU   @ENDDATA-ACYCMS7A                                    00049000
         END   ,(PL/X-370,0103,95125)                               00050000
```

Chapter 3. Overview of CMIP services API functions

VTAM provides a set of API functions for management application programs to use when interfacing with CMIP services. CMIP operations are performed through this interface.

This chapter covers the following topics that relate to the API:
- Decisions an application programmer needs to make before coding
- Requirements for CMIP application programs
- Format of API messages

Decisions to make before coding

You must decide among the following options before you begin coding your application program:

- Do you want to use the common storage area (CSA) interface of the data space interface?
- What form of distinguished name does your application program require from CMIP services?
- Is your application program to be a manager application program or an agent application program?

The following sections describe each of these decisions.

Common storage area storage or data space storage?

The API interface provides either of two mechanisms for receiving messages. These two mechanisms are through the following:
- Common storage area (CSA) interface
- Data space interface

Some differences exist between using CMIP services with the CSA interface and using CMIP services with the data space interface.

Common storage area interface

In the CSA interface, the read queue exit routine is called for each message. Each message is passed in common storage. The CSA interface is intended to be used by low-volume users.

The following exit routines and functions run under the same task:
- Read queue exit routine
- TPEND exit routine, if there is one
- MIBConnect function
- MIBDisconnect function

Figure 1 on page 42 shows the relationship between the application program and CMIP services for an application program using the CSA interface.

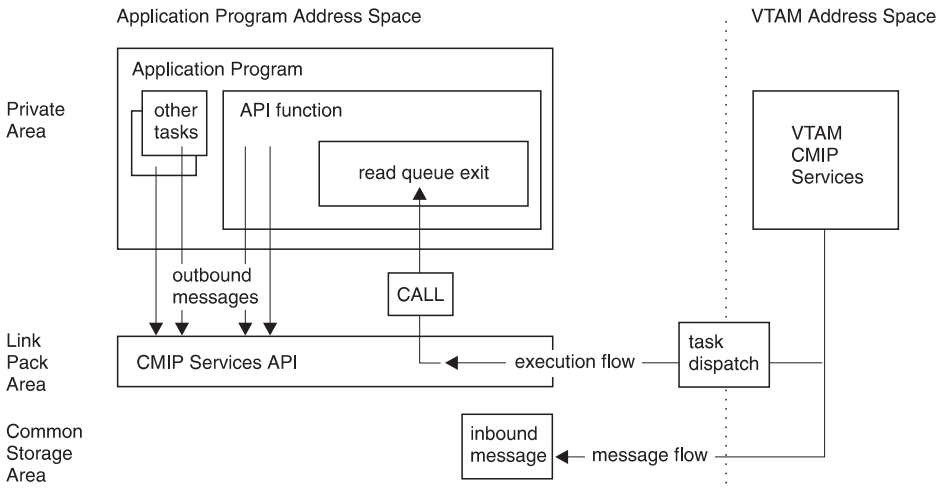


Figure 1. Using CMIP services with the common storage area interface

Data space interface

Application programs that expect to receive a large volume of messages should use the data space interface. For this interface, messages remain in the data space until they are freed by the application program or until the data space fills, whichever occurs first.

The following exit routines and functions run under the same task:

- Read queue exit routine
- TPEND exit routine, if there is one
- MIBConnect function
- MIBDisconnect function

Figure 2 describes the relationship between the application program and CMIP services for an application program using the data space interface.

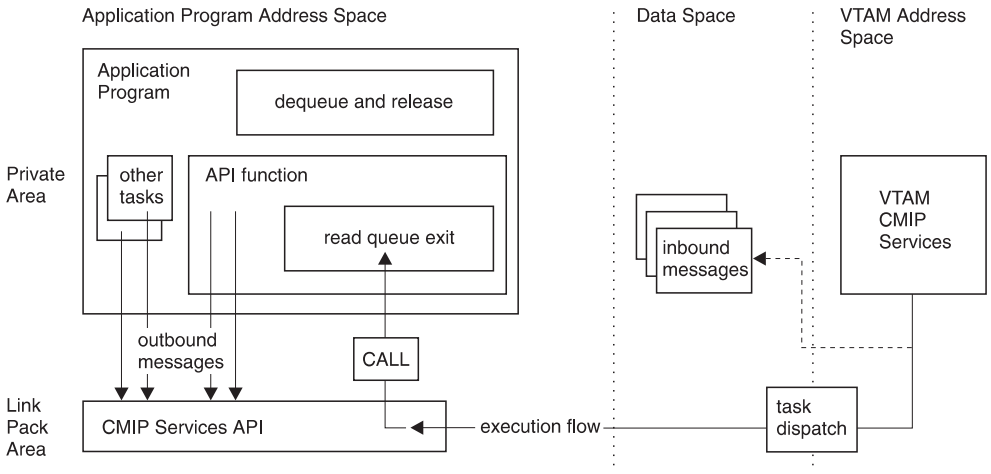


Figure 2. Using CMIP services with the data space interface

An application program that uses one or more of the individual API functions must load the entry point for that function from LPALIB. All modules are placed in LPALIB when the operating system is initialized. Once the entry points for the

APIs are known, the application program can call an API function directly. See Table 2 on page 53 for the module names of each API function. Application programs can have any tasking structure. The functions for reading and freeing messages in the data space are serially reusable only.

Advantages of CSA interface and data space interface

Message strings can be transferred from CMIP services to the application program through either CSA storage or data space storage.

In general, application programs that use the CSA interface are simpler to code. Application programs that use the data space interface are faster.

The data space interface has the following other advantages over the CSA interface:

- Messages can be buffered. They do not have to be retrieved immediately from the data space. With the CSA interface, the application program must copy its message when the read queue exit routine is called. CMIP services frees the CSA storage containing the message on return from the read queue exit routine.
- There are fewer task switches with the data space support. The read queue exit routine is called only when the count of messages waiting in the data space goes from zero to one. The CSA interface, by contrast, calls the read queue exit routine for every message. Each time the read queue exit routine is called, it causes a dispatch of the application program's TCB.
- CSA can be a critical resource in some configurations. The data space interface uses no CSA for inbound messages.

To display the amount of data space storage in use by an application program, use the D NET,STORUSE command. See *z/OS Communications Server: SNA Operation* for more information about this command.

An application program using the data space interface must not allow the messages to back up in the data space to the point where the data space fills. If this occurs, CMIP services stops forwarding messages to the application program until the application program calls the MIBDisconnect function and calls the MIBConnect function again.

Differences between the CSA interface and the data space interface are described throughout this section.

Note: To use data space storage MVS/ESA™ 3.1.3 or higher is required.

The API and the read queue exit routine handle all details of the message flow between the application program and CMIP services. The application program invokes the API when it needs to send a message. CMIP services returns information to the application program according to the following methods:

- **If using the CSA interface,** information is returned by calling the read queue exit routine for each message. For more information about the exit routine, refer to "Read queue exit routine for the CSA interface" on page 88.
- **If using the data space interface,** information is returned by copying each message to the data space and notifying the application program through the read queue exit routine if the number of buffers in the data space goes from zero to one. For more information about the exit routine, refer to "Read queue exit routine for data space storage" on page 89.

Note: When the application program is notified, the application program receives notification again only when the number of buffers returns to zero and goes to one buffer again. The application program must call the routine to dequeue buffers from the data space storage until this routine indicates that there are no more buffers to receive. See "Dequeueing a buffer with the dequeue routine" on page 92 for details.

The read queue exit routine runs in TCB mode in the application program's address space.

What form of distinguished name?

Your application program can choose between two forms of distinguished names: short form and long form. Here is a distinguished name written in short form:

```
1.3.18.0.2.4.6=NETA;2.9.3.2.7.4=(name AAAAAA)
```

Here is the same distinguished name written in long form:

```
(RelativeDistinguishedName (AttributeValueAssertion (attributeType
1.3.18.0.2.4.6, attributeValue "NETA")), RelativeDistinguishedName
 (AttributeValueAssertion (attributeType 2.9.3.2.7.4, attributeVal
ue (name "AAAAAA"))))
```

Application programs can build distinguished names in either form to send to CMIP services. Application programs specify to CMIP services on the connection options parameter passed to the MIBConnect function which form of distinguished names they wish to receive. For a description of the MIBConnect function, refer to "MIBConnect—MIB connection function" on page 56.

What type of application program—manager or agent?

An agent application program represents resources and processes CMIP requests sent to those resources. A manager application program gathers information by sending CMIP requests to resources.

Requirements for CMIP application programs

An application program that uses the API must fulfill the following requirements:

- The API must be called from the home address space.
- The application program must be authorized.
- The application program must use a task mode of the task control block (TCB).

  The read queue exit routine is called under the same TCB that issued the MIBConnect function. An application program with multiple tasks can issue the following API functions from any of its tasks:
  - MIBSendRequest
  - MIBSendResponse
  - MIBSendRegister
  - MIBSendDeleteRegister
  - MIBSendCmipRequest
  - MIBSendCmipResponse

  However, the application program must be prepared to handle the invoking of the read queue exit routine from the task that originated the MIBConnect function.
- The MIBConnect and MIBDisconnect functions must be called from the same task.

- The application program must define the APPL definition statement and specify the name that is to be used on the MIBConnect function. See *z/OS Communications Server: SNA Resource Definition Reference* for more information about the APPL definition statement.
  - A separate APPL definition statement is needed for each MIBConnect function that the application program is expected to perform. The application program cannot call the MIBConnect function again without calling the MIBDisconnect function first.
  - Each successful call to the MIBConnect function that specifies the data space vector parameter causes a new data space to be created. For more information about the MIBConnect function, refer to "MIBConnect—MIB connection function" on page 56.
- No API functions can be issued from the TPEND exit routine or the read queue exit routine.

Calls to API functions can be made from more than one subtask. However, the application program is assumed to be terminated when the subtask that issued the MIBConnect function terminates. When the task that issued the MIBConnect function terminates, the ACB for the application program is closed automatically.

The ACB is not closed automatically if multiple tasks are used and a subtask that meets the following conditions terminates:
- The subtask is using the API.
- The subtask did not open the connection with the MIBConnect function.

Format of API messages

API messages have the following format:

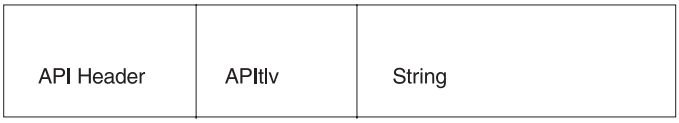


Figure 3. Format of API messages

The type of message is determined by the first field in the API header. The string follows the API header. The syntax of the string includes optional source information, optional destination information, and a required message.

Description and example of the API header

The API header is built for the application program when the application program calls API functions that send messages to CMIP services. It is returned to the application program when the message is sent from CMIP services to the application program.

The API header begins in the first byte of the message. The length of the header varies according to the size of the local identifier. If the message contains data in addition to the API header, the data begins immediately following the API header.

The C language definition of the API header follows. Note that actual local identifiers vary in size from one to eight bytes in length and can be of any data type. It is declared as an eight-character array for simplicity.

Note: To facilitate reading on any host terminal and printing on any host printer, trigraph sequences have been used for square brackets. These sequences are "??(" for left square bracket and "??)" for right square bracket.

```
typedef struct APIhdr_tag
{
  unsigned char msg_type;
  unsigned char api_version;
  unsigned char origin;
  unsigned char RESERVED1;          /* Application programs must not
                                       use or depend on the value of
                                       this field in any way.        */
  unsigned int invokeId;
  unsigned int connectId;
  unsigned int numLocalIds;
  time_t timestamp;
  unsigned short resultCode;
  unsigned char RESERVED2??(2??);   /* Application programs must not
                                       use or depend on the value of
                                       this field in any way.        */
  unsigned int  RESERVED3;          /* Application programs must not
                                       use or depend on the value of
                                       this field in any way.        */
  unsigned char localIds??(8??);
} APIhdr;
```

The actual size of the API header associated with a particular message received from CMIP services is determined by the size of the fixed part (all fields up to but not including the localIDs field) plus the number of attached local identifiers times the size of each local identifier. For this release, the number of attached local identifiers is always one.

The actual size is a useful quantity since the string portions of the message start immediately after the API header.

To make it easier to calculate the actual size, the APIhdrSize macro is provided in the language header file, ACYAPHDH. Given the name of an APIhdr and the size of the application's local identifiers, it returns the actual size of an API header. The following example shows the APIhdrSize macro:

```
#define MY_LOCAL_ID_SIZE 7

APIhdr *APIhdr1;
APIhdr APIhdr2;
size_t Size1, Size2;

Size1 = APIhdrSize(*APIhdr1,MY_LOCAL_ID_SIZE);
Size2 = APIhdrSize( APIhdr2,MY_LOCAL_ID_SIZE);
```

API header fields

A description of each field in the API header follows:

msg_type

Indicates the type and format of message to which this header is attached. An API message can be an indication, a confirmation, or an OSI error. Messages of type API_MSG, API_REG_ACCEPT, API_SVC_COMPLETE, or API_SVC_ERROR contain a formatted string immediately following the API header. This formatted string ends with X'00'.

API_TERMINATE_INSTANCE does not have a string, but X'00' is stored after the local identifier for the convenience of the application program.

CMIP services uses additional values internally for the msg_type field. These values can appear in buffer trace records generated when an application program calls a API function to send data to CMIP services.

Each of the possible msg_type values for APIhdr structure that can be received by an application program is described in the following list.

API_MSG
A CMIP string or a response to a VTAM-specific request or response. An example of a response to a VTAM-specific request is ACF.SubscribeRsp. API_MSG is defined in ACYAPHDH. For a listing, refer to Appendix A, "C language header file (ACYAPHDH)," on page 229.

API_REG_ACCEPT
Sent by CMIP services to indicate that the MIBSendRegister request succeeded. API_REG_ACCEPT is defined in ACYAPHDH. For a listing, refer to Appendix A, "C language header file (ACYAPHDH)," on page 229.

API_SVC_COMPLETE
Sent by CMIP services to indicate that the associated request was processed correctly. This message is returned for unconfirmed CMIP requests. API_SVC_COMPLETE is defined in ACYAPHDH. For a listing, refer to Appendix A, "C language header file (ACYAPHDH)," on page 229.

API_SVC_ERROR
Sent by CMIP services to indicate that the associated request could not be processed. Examples of why it could not be processed are that the string was incorrectly formatted or that there is no network path available to the destination. A specific error code is provided in the message to assist in diagnosing the problem.

In many cases, CMIP services records additional diagnostic information in CMER VIT entry of the VTAM internal trace. See *z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT* for information about the CMER VIT entry. See for *z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures* information about how to use the VTAM internal trace.

API_SVC_ERROR is defined in ACYAPHDH. For a listing, refer to Appendix A, "C language header file (ACYAPHDH)," on page 229.

API_TERMINATE_INSTANCE
Sent by CMIP services to indicate that the object has been deregistered.

API_TERMINATE_INSTANCE is defined in ACYAPHDH. For a listing, refer to Appendix A, "C language header file (ACYAPHDH)," on page 229.

api_version
Reserved for use by CMIP services.

origin
Indicates where the message was generated and how the message should be used. Each of the possible origin field values is described in the following list:

ORIGIN_OBJ
Response to a request that was previously submitted by the object receiving the message. The receiving object can use the invoke identifier to look up the previous request.

ORIGIN_REMOTE
Generated by another object and is a form of unsolicited request or linked

reply. The object receiving this message should use the invoke identifier from the API header and the association data from the string to respond to the message.

invokeId
Can be used to correlate requests and responses. If the origin field is set to ORIGIN_OBJ, the invoke identifier field was generated by the application program when a previous request was sent to CMIP services. If the origin field is set to ORIGIN_REMOTE, the invoke identifier field was generated by a remote object and must be returned in a response along with the association handle so that the remote object can use it for correlation.

connectId
The connect identifier field is reserved for use by the API.

timestamp
Set by the API when a message is sent to CMIP services.

numLocalIds
Specifies the number of local identifiers following the fixed-size portion of the API header. This field is always zero or one.

resultCode
For API_SVC_ERROR messages, the error code is stored here. The same error code also appears in the string.

localIds
Can contain a local identifier. A local identifier is a unique identifier for an object and was provided to MIBSendRegister when that object was registered. If a local identifer is present, it ranges in size from 1 to 8 bytes. The number of bytes is determined by the application program and is specified in a parameter passed to MIBConnect. This local identifiers field is passed back to the application program unchanged by CMIP services.

Description and example of the string

Strings that are included in API_MSG messages begin with the following fields, some of which are optional, depending on whether the API_MSG is a request, indication, response, or confirmation, as shown in Table 1 on page 49.

Table 1. Destination and source fields in string headers

Type of CMIP message	src-type field	src field	dest-type field	dest field
Request	Optional for subtree managers, only. This is choice (2), if included. This is never included for application programs other than subtree managers.	Optional. This is the distinguished name, if included.	Optional. This can be any choice.	Optional. This can be any choice.
Indication	1	assoc-handle	Never included	Never included
Response	Optional for subtree managers, only. This is choice (2), if included. This is never included for application programs other than subtree managers.	Optional. This is the distinguished name, if included.	1	assoc-handle
Confirmation	1	assoc-handle	Never there	Never there

The message field is the only field that must be provided on requests. Responses and linked replies must be formatted with the association data that was provided on the indication. (The indication is the request being answered.) The caller of the MIBSendRequest function or MIBSendResponse function must build the string with all fields. Other API functions do not require the caller to build the string with all fields.

For API functions that build the string automatically, for example, the MIBSendCmipRequest function, separate fields are provided to pass individual fields that are placed in the string by the API function.

The syntax of the string header follows, in ASN.1 notation:

```
Header ::= SEQUENCE
  {
    src-type INTEGER                -- source type
    {
      assoc-handle(1)               -- association handle
    } OPTIONAL,
    src GraphicString OPTIONAL,     -- source
    dest-type INTEGER               -- destination type
    {
      assoc-handle(1),              -- association handle
      full-dn(2),                   -- distinguished name
      ae-title(3                    -- AE title
    } OPTIONAL,
    dest GraphicString OPTIONAL,    -- destination
    msg GraphicString               -- the message itself
  }
```

The format of the required msg field in the string header is dictated by the syntax of the message sent or received by the application program. The following example shows a CMIP string, as received by an application program from CMIP services. This string immediately follows the localIds field of the APIhdr structure.

```
src-type 1, src a1,msg CMIP-1.ROIVapdu (invokeID 327686, operation-v
alue 3, argument (baseManagedObjectClass 1.3.18.0.0.1829, baseManage
dObjectInstance (distinguishedName (RelativeDistinguishedName (Attri
buteValueAssertion (attributeType 1.3.18.0.2.4.6, attributeValue "MY
NETID")), RelativeDistinguishedName (AttributeValueAssertion (attrib
uteType 1.3.18.0.0.2032, attributeValue "MYCPNAME")), RelativeDistin
guishedName (AttributeValueAssertion (attributeType 1.3.18.0.0.1984,
 attributeValue "APPL1")))), synchronization bestEffort, scope (basi
cScope 0), filter (and ()))
```

Rules for the source and destination fields in the string

When messages are received from CMIP services through the API (on indications and confirmations), the following rules apply:

- If the msg_type field in the API header is API_MSG, the src-type field in the string header is set to 1 (assoc-handle) and the src field contains the association handle over which the message arrived.
- If the msg_type field in the API header is not API_MSG, the source data is not present.
- If the local identifier in the API header refers to a subtree manager and the message is not targeted for that subtree manager, the dest-type field in the string is full-dn and the dest field contains the distinguished name of the object instance that is supposed to receive the message.

The application program does not normally build the src-type, src, dest-type, and dest portions of the string, but instead relies on MIBSendCmipRequest and MIBSendCmipResponse functions to build this portion of the string.

The src-type and src fields in the string header need to be provided only if the object needs to override the distinguished name associated with the registered object that is building the message.

The only acceptable src-type is distinguished name (0), which is the default. If the src field is provided and it contains a distinguished name that is different from the provider, the message contains a source override. Only a subtree manager can specify a source name to override the source name in the string header. If an application program that is not a subtree manager specifies a source, the message is flagged with an error.

The dest-type and dest fields are not required. However, these fields can be used to explicitly address messages when the syntax of the message does not contain routing information or when the routing information is not understood by CMIP services. If the CMIP standard is being used, explicit destination information is not required because the destination is given in the managedObjectInstance field.

The same does not apply, however, to OSI responses prompted by an indication and containing the same invoke identifier as the indication. When an object sends a response, it must provide the association handle from the indication that prompted the response.

Messages received by an object instance do not contain the dest-type and dest fields.

The msg field in the string header contains the formatted string. The string must begin with an ASN.1 module name and an ASN.1 syntax name. For all CMIP flows, the module name is CMIP-1 because CMIP-1 is the name of the ASN.1 module that defines the syntaxes used for CMIP flows.

Following the module name is the type name. The module name and type name must be separated by exactly one period; no other characters can be placed between these names. The remainder of the message is defined by the ASN.1 syntax for the module and type specified.

Chapter 4. CMIP services API function syntax and operands

This chapter describes all of the VTAM CMIP services API functions. Function descriptions are arranged alphabetically.

For each function, an example of its use is given. These examples are not intended to show the sequence of operations that an application program must perform as a management application program. They merely show the syntax of calling the API function.

Overview of API functions

Table 2 lists the API functions and indicates the name of the module that must be loaded before invoking an API function. Although logical names such as MIBConnect and MIBSendRegister are used in the table, the physical names of the API functions are the module entry point names.

The abbreviation N/A in the Module Entry Point column indicates that these are the data space modules used for dequeueing or releasing the messages from the data space. The addresses of these modules are returned on the MIBConnect function. Refer to the data space vector format and the interface control block (ISTNMICB) format in the "MIBConnect—MIB connection function" on page 56.

The Type column indicates whether the function is synchronous or asynchronous. For a description of these types, refer to "Synchronous and asychronous functions" on page 55.

Table 2. API functions: module entry point, type, and where to find more information

API function	Module entry point	Type	More information
MIBConnect	ACYAPCNP	Synchronous	Page 56
MIBDisconnect	ACYAPD1P	Synchronous	Page 67
MIBSendRegister	ACYAPRGP	Asynchronous	Page 79
MIBSendDeleteRegistration	ACYAPDRP	Asynchronous	Page 77
MIBSendRequest	ACYAPQRP	Asynchronous	Page 83
MIBSendCmipRequest	ACYAPQCP	Asynchronous	Page 70
MIBSendResponse	ACYAPRSP	Asynchronous	Page 83
MIBSendCmipResponse	ACYAPCPP	Asynchronous	Page 73
Data space dequeue routine	N/A	Synchronous	Page 92
Data space release routine	N/A	Synchronous	Page 93

How the functions are coded

The functions are coded in the same format as C language functions. Parameters are positional, and a value must be specified for each parameter to the function. For some parameters, NULL (a pointer with value zero) may be specified instead of some other value. Such parameters might be described as optional input under the Declarations section for each API function.

Parameters are separated by commas. Parameter values must be specified in the format listed in the declarations section.

For example, in the declarations section of the MIBConnect function, the following line indicates that the API level must be specified as an unsigned integer:

```
      unsigned int,     /* API level - input                          */
```

In the parameter descriptions, the phrase "null-terminated string" means a sequence of EBCDIC characters terminated by a byte containing zero, for example:

```
char *s1 = "Hello";
char s2[6] = {'H','e','l','l', 'o', '\o'}
```

Refer to the appropriate C language publication for your operating system for more information on operand formats.

How the functions are described

For each function, the following information is included:

- Purpose of the function
- Declarations for the function
- Descriptions of the parameters
- A list of return codes
- An example of how the function is coded in an application program

Completion information

If errors occur in CMIP services while processing a request or response, CMIP services sends a MIB.ServiceError message to the object that originated the request or response.

All of the functions have a return code that should be examined by the application program. A value of zero means that the function was successful. Other values alert the application program to incorrect parameters, resource shortages (for example, memory allocation errors), and other problems.

The return codes for each API function are listed under Return codes in the section for each function.

These return codes are used by VTAM CMIP services and appear in the CMER VIT entry and in messages sent from VTAM CMIP services to the application programs. See *z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT* for information about the CMER VIT entry.

Table 3 shows which VIT entries are for each API function.

Table 3. VIT entries for each API function

API function	VIT entry
MIBConnect	MC01 and MC02
MIBDisconnect	MDIS
MIBSendCmipResponse	MQRS
MIBSendCmipRequest	MQRQ
MIBSendDeleteRegistration	MDEL
MIBSendRegister	MREG
MIBSendResponse	MQRS
MIBSendRequest	MQRQ

Synchronous and asychronous functions

The return codes from synchronous functions indicate whether the function completed successfully. MIBConnect and MIBDisconnect are synchronous functions.

The return codes from asynchronous functions indicate only whether CMIP services received the request or response. All API functions, except MIBConnect and MIBDisconnect, are asynchronous functions.

All of the API functions that return an invoke identifier are asynchronous functions. The invoke identifier can be used to correlate the response to the original request. A return code of zero from the API function indicates that the request was successfully sent to CMIP services. The confirmation from the target of the request serves as the acknowledgement.

On confirmed requests, the object sending the request receives a MIB.ServiceError message or a CMIP message (ROIVapdu, RORSapdu, or ROERapdu). On unconfirmed requests, the object sending the request receives a MIB.ServiceAccept message or a MIB.ServiceError message.

Since responses are never confirmed, the object sending the response receives a MIB.ServiceAccept message or a MIB.ServiceError message.

Return codes are integers. A value of 0 always indicates success with no errors to report. The actual confirmation or error report is returned by CMIP services by one of the following methods:

- If using CSA storage, information is returned through the read queue exit routine. See “Read queue exit routine for the CSA interface” on page 88 for details.
- If using data space storage, information is returned by calling the dequeue and release routines returned in the data space vector field. For more information about these routines, refer to Chapter 6, “Dequeue and release routines for data space storage,” on page 91.

MIBConnect—MIB connection function

Purpose

The MIBConnect function returns a link identifier that is used to refer to the connection in future calls to the API.

The MIBConnect function is a synchronous function. The return code from the MIBConnect function indicates whether the function completed successfully.

The MIBConnect function opens an ACB on behalf of the caller. The ACB is closed when the caller calls the MIBDisconnect function or when the task that called the MIBConnect function terminates. The ACB is not closed when CMIP services terminates or when VTAM terminates.

Declarations

The following declarations indicate the order of the parameters for this function.

```
typedef int MIBConnect_t(
      unsigned int,     /* API level - input                        */
      int *,            /* link identifier - output                 */
      unsigned int,     /* maximum outstanding invoke
                           identifiers - input                      */
      const char *,     /* application ACB name - input             */
      void *,           /* TPEND routine pointer - optional input   */
      void *,           /* read queue exit routine pointer - input  */
      unsigned int *,   /* SMAE name buffer size - input/output     */
      char *,           /* SMAE name buffer - output                */
      unsigned int *,   /* System Object name buffer size -
                           input/output                             */
      char *,           /* System Object name buffer - output       */
      int,              /* user data - input                        */
      unsigned int *,   /* OPEN ACB error value - output            */
      char **,          /* VTAM release level - output              */
      const char *,     /* password - optional input                */
      unsigned int,     /* data space vector length - optional input */
      ISTRIV10_t *,     /* data space vector - optional input       */
      unsigned int,     /* local identifier length - input          */
      unsigned int);    /* connection options - input               */
```

Parameters

API level
This parameter must be 0.

link identifier
MIBConnect returns a value in this field. The application program must provide this value in subsequent API calls.

maximum outstanding invoke identifiers
This parameter determines how many unique invoke identifiers can be generated locally by the API. Invoke identifiers are generated on all requests that are sent to CMIP services and can be reused once the response has been received by the requestor. API functions generate and clear invoke identifiers. The caller of the API function does not need to generate or keep track of outstanding invoke identifiers except where needed for its own request/response correlation.

Note: Valid values are 256 to 65535. Input values are changed to meet the minimum or maximum range.

application ACB name
This parameter is a pointer to a null-terminated string that represents the unique name associated with application. The name must be unique among VTAM resources and must be defined on the APPL definition statement. It must follow the naming rules that apply to application programs that open an ACB.

The following APPL definition statement defines TOPOMGR as the application program's ACB name.

```
TOPOMGR  APPL ACBNAME=TOPOMGR
```

See *z/OS Communications Server: SNA Resource Definition Reference* for information about the ACBNAME operand on the APPL definition statement.

Note: The value of this parameter is converted to uppercase before being passed to OPEN ACB.

TPEND routine pointer
This parameter is the address of an application assembler routine to be called by VTAM if VTAM terminates before the application program terminates or issues the MIBDisconnect function. Specify NULL if you do not wish to provide a termination exit routine.

See *z/OS Communications Server: SNA Programming* for information about the TPEND exit routine.

As with other TPEND exit routines, the application program should clean up in an orderly manner for a normal HALT command. The application program should deregister objects, discard EFDs, and disconnect.

In response to a HALT QUICK or HALT CANCEL command, the application program should not attempt to clean up. It should only issue the MIBDisconnect function.

Note: The ACBUSER field are set to the value of the user data parameter supplied on the MIBConnect function when the TPEND exit routine is scheduled.

read queue exit routine pointer
This parameter is the address of an application assembler routine to be called by CMIP services when messages are to be received. See Chapter 5, "Read queue exit routine," on page 87 for information about the read queue exit.

SMAE name buffer size
This is the size of the buffer provided by the application for the SMAE name. 100 bytes is the recommended size for this buffer.

MIBConnect is set the value to the actual length (including the terminating zero) on output.

If the buffer provided is not long enough, MIBConnect returns the MB_ERR_STORAGE_TOO_SMALL return code. The application should allocate a new buffer using the updated value of this parameter and call MIBConnect again.

SMAE name buffer
MIBConnect places a pattern for building SMAE names in the storage pointed to by this parameter. The application program can use this pattern with the C Standard Library function **sprintf** to build the name of a SMAE name on this host.

The following example SMAE name format as returned by MIBConnect:

```
1.3.18.0.2.4.6=NETA;2.9.3.2.7.4=(name SSCP1A);1.3.18.0.2.4.12=%s
```

Assuming that format is the name of a character array containing the format string and AE is the name of a character array to hold the resulting SMAE name, the following code will build a SMAE name that could be used for the RegisterAE special CMIP Services request.

```
sprintf(AE,format,"MyApplName");
```

The name of the default SMAE provided by CMIP Services has OSISMASE as the final attribute value in the distinguished name.

System Object name buffer size
This is the size of the buffer provided by the application for the System Object. The recommended size for this buffer is 100 bytes.

MIBConnect sets the value to the actual length (including the terminating zero) on output.

If the buffer provided is not long enough, MIBConnect returns MB_ERR_STORAGE_TOO_SMALL. The application program should then allocate a new buffer using the updated value of this parameter and call MIBConnect again.

System Object name buffer
MIBConnect places the name of the System Object into this buffer.

The System Object name should be used when creating local EFDs; EFDs are named “under” the System Object.

user data
This four-byte field is provided to the application program on entry to the read queue and to the TPEND exit routines.

OPEN ACB error value
When control is returned to the application program and the return code is MB_ERR_CONNECT, the OPEN ACB error value parameter needs to be evaluated.

The following list shows the OPEN ACB error values returned in the OPEN ACB error value parameter.

ERROR Field
Meaning

0 (X'00')
OPEN successfully opened this ACB.

4 (X'04')
The ACB has been opened.

20 (X'14')
OPEN cannot be processed because of a temporary shortage of storage.

36 (X'24')
The OPEN ACB failed for one of the following reasons:

- The password specified by the ACB did not match the corresponding password in the APPL entry.
- The ACB did not specify a password and the APPL contains one.
- The security management product determined that the user is not authorized to open the ACB.

70 (X'46')
OPEN was issued in an exit routine.

80 (X'50')
VTAM has not been included as part of the operating system. The fault lies in the system definition procedures.

82 (X'52')
VTAM is included as part of the operating system, but the VTAM operator issued a HALT command, and VTAM has shut down. You cannot attempt to establish a session or communicate with any LUs.

84 (X'54')
Either the address supplied in the ACB's APPLID field lies beyond the addressable range of your application program, or no entry is found in the VTAM configuration tables that matches the name indicated by the ACB's APPLID field (or supplied by the operating system). If the OPEN macroinstruction is specified correctly, your system programmer might have:

- Failed to include your application program's symbolic name during VTAM definition
- Improperly handled the symbolic name

Refer to the description of the APPLID operand in the ACB macroinstruction.

86 (X'56')
A match for your application program's symbolic name is found, but it is for an entry other than an APPL. If you specified this name in the ACB's APPLID field, verify that you have the correct name and handled this name properly (see the APPLID operand of the ACB macroinstruction). If the symbolic name is supplied by the operating system, the supplied name is suspect.

88 (X'58')
Another ACB, already opened by VTAM, indicates the same application program symbolic name that this ACB does. The system programmer might have assigned the same symbolic name to two application programs. This is valid only if the programs are not open concurrently. Possibly the system operator initiated your program at the wrong time.

90 (X'5A')
No entry is found in the VTAM configuration tables that matches the name indicated by the ACB's APPLID field (or supplied by the operating system). This error might have occurred for one of the following reasons:

- The VTAM operator deactivated the APPL entry.
- The APPL entry was never created.
- If VTAM is trying to recover for persistent sessions, the application is not in pending recovery state.

92 (X'5C')
VTAM is included as part of the operating system but inactive.

94 (X'5E')
The address supplied in the ACB's APPLID field lies beyond the addressable range of your application program.

95 (X'5F')
The VTAM transient being used by the application for an OPEN ACB does not match the level of VTAM.

96 (X'60')
An apparent system error occurred. Either there is a logic error in VTAM, or there is an error in your use of OPEN or CLOSE that VTAM did not properly detect. Save all applicable program listings and storage dumps, and consult IBM Service.

98 (X'62')
The APPLID length byte is incorrectly specified.

100 (X'64')
The address supplied in the ACB's PASSWD field lies beyond the addressable range of your application program.

102 (X'66')
The PASSWD length byte is incorrectly specified.

104 (X'68')
The APPLID field in the ACB identifies an application program that is defined with AUTH=PPO in its APPL definition statement. Another program with the same authorization is active. Only one program defined with AUTH=PPO can be active at a time.

106 (X'6A')
The address supplied in the ACB's vector list field lies beyond the addressable range of your application program.

108 (X'6C')
The VTAM ACB vector list length byte is incorrectly specified.

112 (X'70')
You attempted to open an ACB that is in the process of being closed. This can occur when a VTAM application program job step or subtask is canceled or terminates abnormally. The process of closing the ACB can continue after the job step or subtask has actually terminated. Subsequently, if the job step is restarted or the subtask is reattached before the ACB closing process has been completed, an OPEN macroinstruction that is then issued for that ACB fails.

114 (X'72')
This code occurs when an OPEN ACB fails for an LU 6.2 application with VERIFY=OPTIONAL or VERIFY=REQUIRED for one of the following reasons:

- The security management product is not installed.
- The security management product is not active.
- The security management product resource class APPCLU is not active.
- The application represented by the ACB is not in the security management product Started Procedures Table.

116 (X'74')
VTAM rejected the takeover by an alternate application because the original application did not enable persistence, although it is capable of persistence.

118 (X'76')
OPEN failed because the specified application is in a recovery pending

state and PERSIST=YES is not specified on the ACB that is being opened. The OPEN may also fail if the application is in pending terminate state and an active CDRSC with the same name has been found in the sysplex.

120 (X'78')
ACB option mismatch between original application and opening takeover or recovery application. One or more of the following can apply:

- MACRF mismatch—both values must be either LOGON or NLOGON; they cannot differ.
- NQNAMES mismatch—both applications must be specified as NQNAMES=YES or NQNAMES=NO; they cannot differ.
- PERSIST mismatch—both applications must be specified as PERSIST=YES.
- FDX mismatch—both applications must be specified as FDX=YES or FDX=NO; they cannot differ.
- ISTVAC81 mismatch—the application capabilities vector provided by the recovering application does not match that of the original application.

140 (X'8C')
PERFMON=YES is coded on the ACB but the application is not CNM and POA authorized.

188 (X'BC')
The ACB is in the process of being opened or closed by another request.

244 (X'F4')
The application program is not authorized for SRBEXIT=YES. A request to open an ACB whose corresponding APPL definition statement specifies SRBEXIT=YES is rejected unless the application program is APF authorized, or using key 0–7, or in supervisor state.

246 (X'F6')
NIB storage address not valid. A CNM authorized application program either failed to supply an NIB pointer in the NIB field of the ACB, or the NIB address supplied lies beyond the addressable range of the application program.

250 (X'FA')
NIB options not valid. Either an application program without CNM authorization (specified in its associated VTAM resource definition) supplied an NIB pointer in its ACB; or, if CNM authorized, the application program failed to supply valid NIB options on the NIB macroinstruction.

254 (X'FE')
Duplicate unsolicited RU routing requested. The CNM routing table indicated that this application program was to receive the same unsolicited formatted requests that were already being routed to another active CNM authorized application program. Only one application program can be actively receiving a particular type of RU (for example, RECFMS) at a time.

VTAM release level
Indicates the address of VTAM release-level vector. See *z/OS Communications Server: SNA Programming* for more information about the format of this vector.

password
Specifies a pointer to a null-terminated string. The application program should specify NULL if no password is to be supplied. If a password is specified on PRTCT operand of the APPL definition statement, MIBConnect fails unless a matching password is provided in the password parameter. Password protection is to prevent a program from running as a predefined application program without authorization.

The value of the password is specified on the PRTCT operand of the APPL definition statement. The value must conform to the rules for coding this operand described in the *z/OS Communications Server: SNA Resource Definition Reference*. The maximum length is 8 bytes. Valid passwords contain only alphanumeric characters.

If application program's ACB name is TOPOMGR, the APPL definition statement with a password is similar to the following example:

```
TOPOMGR  APPL  ACBNAME=TOPOMGR,PRTCT=password
```

Note: The value of this parameter is converted to uppercase before being passed to OPEN ACB. This is because VTAM converts the related definition to uppercase but does not convert OPEN ACB parameters to uppercase. Without the conversion to uppercase by MIBConnect, this function would fail if the application provided a lowercase value for this parameter.

data space vector length
If using data space storage, specify a value that is at least the size of (ISTRIV10_t), which is the length of the data space vector. If you are using common storage area storage, specify 0. For an explanation of these types of storage, see "Common storage area storage or data space storage?" on page 41.

data space vector
If you are using data space storage, specify the address of the data space vector (ISTRIV10_t). If you are using the CSA interface, specify NULL. If the MIBConnect function is successful, the fields in this control block are set by VTAM.

The format of the data space vector is:

Offset Meaning

0 (X'00')
Vector Length

1 (X'01')
Vector identifier = X'10'

2 (X'02')
Name of data space. (The field is 8 bytes long.)

10 (X'0A')
Address of interface control block (ISTNMICB)

14 (X'0E')
STOKEN for data space interface. This value is used in ALESERV MVS macro to obtain the ALET value.

22 (X'16')
Reserved

26 (X'1A')
Address of the dequeue routine

30 (X'1E')
Address of the release routine

The ISTNMICB structure is allocated in the data space. The user application must copy this structure into private storage for any future references because the data space can be deleted at any time if VTAM is terminated. Referring to the original after the data space has been deleted results in an abend. By contrast, calling the dequeue and release routines using private copies of their addresses causes an error indication to be returned. It is not valid to refer directly to the data space through a means other than the dequeue or release routine. Those routines should not be called after VTAM is terminated or after issuing the MBDisconnect function.

The format of the interface control block (ISTNMICB) is:

Offset Meaning

0 (X'00')
Reserved

4 (X'04')
Address of the dequeue routine

8 (X'08')
Address of the release routine

local identifier length
Indicates the size of the local identifiers for this application program. The range is 1—8.

connection options
Specify one of the following values:

NO_CONNECT_OPTIONS
Indicates that the application program is to use default behaviors for the connection with CMIP services.

SHORT_NAMES
Indicates that CMIP services is to send distinguished names to the application program in the short form. Otherwise, CMIP services sends distinguished names to the application program in the long form. In either case, the application program can format distinguished names in strings sent through the API functions in either short or long form.

For a description of short and long names, refer to "What form of distinguished name?" on page 44.

Return codes

0 The MIBConnect function was successful, but warning messages might have been issued. Check the OPEN ACB error value parameter for warning messages. See the list of OPEN ACB error values on page 58.

MB_ERR_ALLOC
An error occurred allocating storage. If MB_ERR_ALLOC is received by the application program from an API function and there is a corresponding REQS record in the VIT with a nonzero return code, the LPBUF pool is not large enough and should be increased.

MB_ERR_CMIP_SERVICES_INACTIVE
CMIP services is inactive.

If using common storage area storage, the read queue exit routine stops functioning.

If using data space storage, messages are not put on the data space.

MB_ERR_CONNECT
The MIBConnect was not successful. If the error condition indicated by the OPEN ACB error value parameter can be eliminated, another MIBConnect can be issued.

MB_ERR_INVALID_DS_VECTOR
The value specified for the data space vector length parameter is valid, but the data space vector parameter is not provided.

MB_ERR_INVALID_API_LEVEL
An incorrect value for the API level parameter was passed.

MB_ERR_INVALID_APPL_NAME
The value specified for the application name parameter is longer than 8 characters.

MB_ERR_INVALID_CONNECT_OPTIONS
The value specified on the connection options parameter is not valid. Specify either NO_CONNECT_OPTIONS or SHORT_NAMES as the value for the connection options parameter.

MB_ERR_INVALID_DS_VECTOR_SIZE
If the data space vector parameter is specified, the data space vector length must be at least the size of (ISTRIV10_t), which is the length of the data space vector.

MB_ERR_INVALID_ENVIRONMENT
Data space storage was specified on the data space vector length parameter, but the environment does not support data spaces.

MB_ERR_INVALID_ERROR_FLAG
The OPEN ACB error value parameter does not point to a valid storage location.

MB_ERR_INVALID_LINK_ID
The value specified on the link identifier parameter does not refer to a valid connection.

MB_ERR_INVALID_LOCAL_ID_SIZE
The value specified on the local identifier length parameter is outside the acceptable range of 1—8.

MB_ERR_INVALID_MAX_INVOKE_IDS
The value specified for the maximum outstanding requests parameter is not valid.

MB_ERR_INVALID_PASSWORD
The value specified for the password parameter is not between 0 and 8 characters.

MB_ERR_INVALID_READ_QUEUE_EXIT
The read queue exit routine was not provided.

MB_ERR_INVALID_RELEASE_LEVEL
The value specified for the VTAM release level parameter is not valid.

MB_ERR_INVALID_SMAE_NAME
The value specified for the SMAE name buffer parameter is not valid.

MB_ERR_INVALID_SMAE_NAME_SIZE
The buffer sent to the MIBConnect function is too small to accommodate the name of the SMAE. The actual amount of storage required is returned in the SMAE name length parameter.

MB_ERR_INVALID_SYSTEM_NAME
The value specified for the system object name buffer parameter is not valid.

MB_ERR_INVALID_SYSTEM_NAME_SIZE
The buffer sent to the MIBConnect function is too small to accommodate the name of the system object. The actual amount of storage required is returned in the system object name buffer size parameter.

MB_ERR_INVALID_TPEND_EXIT
The TPEND exit routine is not valid.

MB_ERR_INVALID_USER_DATA
The user data parameter was not provided.

MB_ERR_VTAM_INACTIVE
VTAM is inactive.

Example of function in an application program

The following example shows how the MIBConnect function can be coded in an application program.

```
typedef struct ReadQueueExitData_tag
{
  int ECB;
  int ReasonCode;
  char Buffer ??(16384??);
} ReadQueueExitData_t;

typedef void *LocalId_t;

char SMAE                ??(100??);
char SystemObject        ??(100??);
char *VTAM_Release;
const char *ApplName;
const char *Password;
int LinkId;
int rc;
ReadQueueExitData_t ReadQueueExitData;
size_t SMAE_Size, SystemObjectSize;
unsigned int ACB_Info;
extern void ACYCMS2A(void);
extern void ACYCMS6A(void);

rc = APIs.MIBConnect(0,         /* always zero for this release  */
              &LinkId,          /* MIBConnect will fill in LinkId
                                   with a handle to the
                                   connection.                   */
              65536,            /* maximum number of outstanding
                                   requests                      */
              ApplName,         /* ACB name                      */
              (void *)ACYCMS6A, /* TPEND exit                    */
              (void *)ACYCMS2A, /* address of the Read
                                   Queue Exit                    */
              &SMAE_Size,       /* On input, this is the size of
                                   the SMAE buffer.  On output,
                                   this is the length of the SMAE
                                   name.                         */
              SMAE,             /* This is where MIBConnect will
                                   store the SMAE name (if there
```

```
                    is enough room).              */
&SystemObjectSize,/* On input, this is the
                    size of the System Object name
                    buffer.  On output, this is the
                    length of the System Object
                    name.                         */
SystemObject,     /* This is where MIBConnect will
                    store the System Object name
                    (if there is enough room).    */
(int)&ReadQueueExitData, /* This will be provided
                    to this application's read
                    queue exit by CMIP Services.  */
&ACB_Info, /* If an error occurs opening the
              ACB, this will contain the
              OPEN ACB error code.          */
&VTAM_Release, /* MIBConnect will store the
              address of the VTAM release
              level here.                   */
Password,  /* ACB password                  */
0,         /* dataspace not used            */
NULL,      /* dataspace not used            */
sizeof(LocalId_t), /* size of local ids
              for all objects registered by
              this application              */
0);        /* no special options specified  */
```

MIBDisconnect—MIB disconnection function

Purpose

The MIB disconnection function sends a message to the API to terminate the session and clear all outstanding requests on the connection. There might be many objects registered under one MIB connection and all outstanding requests for those objects are cleared by the MIB disconnect service. The application program using the CMIP services connection should not terminate the connection unless all outstanding requests might be lost without damage to the registered objects.

MIBDisconnect is a synchronous service. The return code from the MIBDisconnect function indicates whether the function completed successfully.

If you want to call a MIBConnect function with the same application ACB name as the ACB name used on a previous MIBConnect function, you must call the MIBDisconnect function before calling the MIBConnect function.

The MIBConnect function opens an ACB on behalf of the caller. The ACB is closed when the caller calls the MIBDisconnect function or when the task that called the MIBConnect function terminates. The ACB is not closed when CMIP services terminates or when VTAM terminates.

If using data space storage, the data space is freed by VTAM. The application program must not call the data space storage dequeue and release routines after it calls the MIBDisconnect function, because the results are unpredictable and the application program might abend.

Declarations

The following declarations indicate the order of the parameters for this function.

```
typedef int MIBDisconnect_t(
                int,              /* link identifier - input        */
                unsigned int *);  /* CLOSE ACB error value - output */
```

Parameters

link identifier
Specifies the link identifier returned by the MIBConnect function.

CLOSE ACB error value
When control is returned to the application program and the return code is MB_ERR_MIBDISCONNECT, this flag needs to be evaluated.

The following list shows the CLOSE ACB error values returned in the CLOSE ACB error value parameter.

ERROR Field
Meaning

0 (X'00')
CLOSE successfully closed the ACB.

4 (X'04')
A CLOSE macroinstruction has been successfully issued for this ACB (or the ACB has never been opened in the first place).

20 (X'14')
CLOSE cannot be processed because of a temporary shortage of storage.

64 (X'40')
Outstanding OPNDST OPTCD=ACQUIRE is not released.

66 (X'42')
The ACB has been closed, but an apparent system error has prevented the successful termination of one or more of the sessions that the application program has. There is a logic error in VTAM; consult IBM Service. The LUs that have not had their sessions terminated are not available to other application programs, and LUs with which you were requesting a session when the CLOSE macroinstruction was issued are likewise unavailable. You can notify the VTAM operator (while the program is running) of the situation so that the operator can make the LUs available to other application programs.

70 (X'46')
CLOSE was not issued in the mainline program. OPEN and CLOSE cannot be issued in any exit routine.

76 (X'4C')
This application program is authorized to issue VTAM operator commands and receive VTAM messages. A CLOSE was issued, but messages are still queued for it, or VTAM is waiting for a reply, or both. See *z/OS Communications Server: SNA Programming* for more information.

80 (X'50')
VTAM is no longer included as part of the operating system.

96 (X'60')
An apparent system error occurred. Either there is a logic error in VTAM; or there is an error in your use of OPEN or CLOSE that VTAM did not properly detect. Save all applicable program listings and storage dumps, and consult IBM Service.

112 (X'70')
CLOSE was issued while the program was in the process of terminating abnormally. The CLOSE is not necessary because the ACB is closed by VTAM when the task terminates.

188 (X'BC')
The ACB is in the process of being opened or is in the process of being closed by another request.

Return codes

0 The MIBDisconnect was successful, but warning messages might have been issued. Check the CLOSE ACB error value parameter for warning messages. See the list of CLOSE ACB error values on page 67.

MB_ERR_MIBDISCONNECT
The MIBDisconnect function was not successful. If the error condition indicated by the CLOSE ACB error value parameter can be eliminated, another MIBDisconnect can be issued.

MB_ERR_INVALID_LINK_ID
The value specified on the link identifier parameter does not refer to a valid connection.

MB_ERR_ALLOC
An error occurred allocating storage. If MB_ERR_ALLOC is received by the application program from an API function and there is a corresponding

REQS record in the VIT with a nonzero return code, the LPBUF pool is not large enough and should be increased.

MB_ERR_INVALID_ERROR_FLAG
The CLOSE ACB error value parameter does not point to a valid storage location.

Example of function in an application program

The following example shows how the MIBDisconnect function can be coded in an application program.

```
int LinkId;
int rc;
unsigned int ACB_Info;

rc = APIs.MIBDisconnect(LinkId, /* This is the handle returned by
                                   MIBDisconnect.               */
                   &ACB_Info);  /* If an error occurs closing the
                                   ACB, MIBDisconnect will store
                                   the CLOSE ACB error code here.*/
```

MIBSendCmipRequest—CMIP request function

Purpose

Use this function when an application program or object is sending a CMIP request.

The MIBSendCmipRequest function queues a request to CMIP services. Use the MIBSendCmipRequest function for CMIP requests instead of the MIBSendRequest function, to allow consistent manipulation of messages.

Declarations

The following declarations indicate the order of the parameters for this function.

```
typedef int MIBSendCmipRequest_t(
              int,                /* link identifier - input       */
              unsigned int,       /* argument type - input         */
              const char *,       /* argument - input              */
              const void *,       /* local identifier - input      */
              const char *,       /* source - input                */
              unsigned int,       /* destination type - input      */
              const char *,       /* destination - input           */
              unsigned int *);    /* returned invoke identifier -
                                       output                       */
```

Parameters

link identifier
Specifies the link identifier returned by the MIBConnect function.

argument type
This should be the CMIP operation value of the operation being requested.

The operation values are given in ACYIDCMS.

argument
This null-terminated string contains the bulk of the request. The ASN.1 type is determined by the CMIP operation value of the request, and is found in the ANY DEFINED BY table for the operation value in ACYIDCMS.

local identifier
Pointer to the local identifier of the object that generated this request. The same local identifier appears in a subsequent response.

source
The distinguished name of the originator of the request. This can be used to override the source of the message. This is used to resolve any appearance of the MIB variable distinguished name. Specify NULL if you do not choose to specify a value.

destination type
This specifies the type of destination data that is being proved in the destination argument. The valid values are DS_NOT_PROVIDED, DS_FULL_DN, DS_ASSOC_HANDLE, and DS_AE_TITLE.

If this field is set to DS_NOT_PROVIDED, then the stack uses the object name in the CMIP parameter as the destination object name.

destination
This specifies the destination of a CMIP string. Specify NULL if the destination type parameter is DS_NOT_PROVIDED. Otherwise, specify the pointer to a distinguished name, association handle, or application entity title.

returned invoke identifier
Specifies the invoke identifier. The invoke identifier is used to correlate this request with a response that arrives subsequently.

Return codes

0 The function was successful.

MB_ERR_ALLOC
An error occurred allocating storage. If MB_ERR_ALLOC is received by the application program from an API function and there is a corresponding REQS record in the VIT with a nonzero return code, the LPBUF pool is not large enough and should be increased.

MB_ERR_INVALID_ARGUMENT
The argument parameter was not provided.

MB_ERR_INVALID_ARGUMENT_TYPE
An incorrect argument type parameter was provided.

MB_ERR_CMIP_SERVICES_INACTIVE
CMIP services is inactive.

If using common storage area storage, the read queue exit routine stops functioning.

If using data space storage, messages are not put on the data space.

MB_WARN_DATA_SPACE_FULL
If using a data space and the data space is out of storage, this warning is returned to remind the application program that no messages will be returned to this application program. This message will still be routed to CMIP services.

MB_ERR_INVALID_DEST
The value of the destination parameter is inconsistent with the value of the destination type parameter. This return code is returned if, for example, destination type is DS_ASSOC_HANDLE, but destination is NULL.

MB_ERR_INVALID_DEST_TYPE
An incorrect destination type parameter was passed.

MB_WARN_EXIT_FAILURE
If using common storage area storage and the application program has indicated that it has had an unrecoverable error when returning to the read queue exit routine, this warning is returned to remind the application program that no messages will be returned to the application program. This message will still be routed to CMIP services.

MB_ERR_INVALID_LINK_ID
The value specified on the link identifier parameter does not refer to a valid connection.

MB_ERR_INVALID_INVOKE_ID
The invoke identifier parameter was not provided.

MB_ERR_LOCAL_ID_MISSING
A local identifier was not provided.

MB_ERR_INVALID_MAX_INVOKE_IDS
The value specified for the maximum outstanding requests parameter is not valid.

MB_ERR_MSG_MISSING
The message parameter was not provided.

MB_ERR_TRANSMIT
An apparent error occurred. Either there is a logic error in VTAM, or the MIBDisconnect function has been issued, but it has not completed.

MB_ERR_VTAM_INACTIVE
VTAM is inactive.

Example of function in an application program

The following example shows how the MIBSendCmipRequest function can be coded in an application program.

```
char CMIP_StringArgument ??(512??);
int LinkId;
int rc;
LocalId_t *MyObjectId;
unsigned int InvokeId;

rc = APIs.MIBSendCmipRequest(LinkId, /* handle returned by
                                        MIBConnect                    */
                             3,      /* operation value is GET        */
                             CMIP_StringArgument,
                             &MyObjectId,
                             NULL,
                             DS_NOT_PROVIDED,
                             NULL,
                             &InvokeId);
```

MIBSendCmipResponse—CMIP response function

Purpose

Use this function when an application program or object is sending a CMIP response. MIBSendCmipResponse queues responses to CMIP services associated with requests that were previously received by the application program from CMIP services.

Declarations

The following declarations indicate the order of the parameters for this function.

```
typedef int MIBSendCmipResponse_t(
               int,                 /* link identifier - input       */
               unsigned int,        /* invoke identifier - input     */
               unsigned int,        /* last in chain - input         */
               unsigned int,        /* success - input               */
               unsigned int,        /* argument type - input         */
               const char *,        /* argument - input              */
               const void *,        /* local identifier input        */
               const char *,        /* source - input                */
               const char *,        /* association handle - input    */
               unsigned int *);     /* returned invoke identifier -
                                         output                      */
```

Parameters

link identifier
Specifies the link identifier returned by the MIBConnect function.

invoke identifier
This is the invoke identifier of the request which is being responded to with this API call.

last in chain
This indicates to CMIP services whether this message is the last response that is generated by this application program for this request. This allows CMIP services to construct the correct message (linked reply or response). A nonzero value indicates that the response is the last in a chain of responses (RORSapdu or ROERapdu). A zero value indicates that the response is not the last in a chain of responses (ROIVapdu—linked reply).

success
This indicates whether the response is positive or negative. This indicates to CMIP services how to interpret the next parameter. A nonzero value indicates that the response represents a positive, successful response. A zero value indicates that the response is negative.

Note: If the last in chain parameter is zero, the success parameter must be nonzero. A linked reply cannot be sent as an error.

argument type
For linked-replies (messages with the last in chain parameter set to zero), this should be two, the CMIP operation value for a linked-reply.

For RORSapdu messages, this should be the CMIP operation value of the operation being responded to.

For ROERapdu messages, this should be the CMIP error value.

The operation values and error values are given in ACYIDCMS.

argument
This null-terminated string contains the bulk of the CMIP string which is built by CMIP services, on behalf of the application program, for this API function.

For ROIVapdu messages (when the last in chain parameter is zero), this string is used for the value of the argument parameter.

For RORSapdu messages (when the last in chain parameter is nonzero and the success parameter is nonzero), this string is used for the value of the result parameter.

For ROERapdu messages (when the last in chain parameter is nonzero and the success parameter is zero), this string is used for the value of the parameter.

local identifier
Pointer to the local identifier of the object that is responding. Specify the same identifier as the one specified in the request.

source
The distinguished name of the originator of the request. This can be used to override the source of the message. This is used to resolve any appearance of the MIB variable distinguished name. Specify NULL if you do not choose to specify a value.

association handle
This is the association identifier of the association that is to be used to send the response. It is required and must be the same as the association handle that was received on the message that is being answered.

returned invoke identifier
Specifies the invoke identifier. The invoke identifier is used to correlate this request with a response that arrives subsequently. This will be filled in only for linked replies. For linked replies, the last in chain parameter is zero.

Return codes

0 The function was successful.

MB_ERR_ALLOC
An error occurred allocating storage. If MB_ERR_ALLOC is received by the application program from an API function and there is a corresponding REQS record in the VIT with a nonzero return code, the LPBUF pool is not large enough and should be increased.

MB_ERR_ARGUMENT_MISSING
The argument parameter was not provided.

MB_ERR_ARGUMENT_TYPE_MISSING
An argument type parameter was not provided.

MB_ERR_ARGUMENT_TYPE_INVALID
An incorrect argument type parameter was provided.

MB_ERR_ASSOC_HANDLE_MISSING
The association handle parameter was not provided.

MB_ERR_CMIP_SERVICES_INACTIVE
CMIP services is inactive.

If using common storage area storage, the read queue exit routine stops functioning.

If using data space storage, messages are not put on the data space.

MB_WARN_DATA_SPACE_FULL
If using a data space and the data space is out of storage, this warning is returned to remind the application program that no messages will be returned to this application program. This message will still be routed to CMIP services.

MB_ERR_DEST_TYPE_INVALID
An incorrect destination type parameter was passed.

MB_WARN_EXIT_FAILURE
If using common storage area storage and the application program has indicated that it has had an unrecoverable error when returning to the read queue exit routine, this warning is returned to remind the application program that no messages will be returned to the application program. This message will still be routed to CMIP services.

MB_ERR_INVALID_LINK_ID
The value specified on the link identifier parameter does not refer to a valid connection.

MB_ERR_INVOKEID_MISSING
The invoke identifier parameter was not provided.

MB_ERR_LAST_IN_CHAIN_MISSING
The last in chain parameter was not provided.

MB_ERR_LOCAL_ID_MISSING
A local identifier was not provided.

MB_ERR_MAX_OUTSTANDING
The value specified for the maximum outstanding requests parameter is not valid.

MB_ERR_SUCCESS_MISSING
The success argument parameter was not provided.

MB_ERR_TRANSMIT
An apparent error occurred. Either there is a logic error in VTAM, or the MIBDisconnect function has been issued, but it has not completed.

MB_ERR_VTAM_INACTIVE
VTAM is inactive.

Example of function in an application program

The following example shows how the MIBSendCmipResponse function can be coded in an application program.

```
#include "acyaphdh"

#define FALSE 0
#define TRUE  1

extern void *MyLocalId_ptr;
int rc;
int LinkId;
unsigned int InvokeId;
MIBSendCmipResponse_t *MIBSendCmipResponse;

/*******************************/
/* Send accessDenied ROERapdu. */
/*******************************/

rc = MIBSendCmipResponse(LinkId,
               InvokeId,    /* the invoke identifier from the request */
```

```
TRUE,        /* last in chain (not linked reply)       */
FALSE,       /* not successful (i.e., ROERapdu)        */
7,
"(invokeID 1179660, error-value 2)",
MyLocalId_ptr,
NULL,
"a1",        /* association handle of the request
                being answered                          */
NULL);       /* no new invoke identifier since
                last-in-chain is TRUE                   */
```

MIBSendDeleteRegistration—Deregistration function

Purpose

The MIBSendDeleteRegistration deletes a registered object. Any objects registered under the object being deleted are also deleted. An object's registration can be removed by local identifier or by distinguished name. Only one of them is required. Both can be provided.

A non-NULL value in the distinguished name parameter indicates that a valid distinguished name was provided.

Declarations

The following declarations indicate the order of the parameters for this function.

```
typedef int MIBSendDeleteRegistration_t(
               int,                 /* link identifier - input       */
               unsigned int *,      /* returned invoke
                                       identifier - output           */
               const void *,        /* local identifier - optional
                                       input                         */
               const char *);       /* distinguished name - optional
                                       input                         */
```

Parameters

link identifier
Specifies the link identifier returned by the MIBConnect function.

returned invoke identifier
Specifies the invoke identifier. The invoke identifier is used to correlate this request with a response that arrives subsequently.

local identifier
Pointer to the local identifier of the object that is to be deleted. Specify NULL for the local identifier parameter if only the distinguished name is provided.

distinguished name
This is the distinguished name of the object instance being deleted. If you provide a local identifier, the distinguished name is optional. Specify NULL if you do not provide a distinguished name.

If you specify a name for this parameter, CMIP services uses the name to look up the object instance to be deleted or to verify that the object instance selected with the local identifier has a matching name.

Return codes

0 The function was successful.

MB_ERR_ALLOC
An error occurred allocating storage. If MB_ERR_ALLOC is received by the application program from an API function and there is a corresponding REQS record in the VIT with a nonzero return code, the LPBUF pool is not large enough and should be increased.

MB_ERR_CMIP_SERVICES_INACTIVE
CMIP services is inactive.

If using common storage area storage, the read queue exit routine stops functioning.

If using data space storage, messages are not put on the data space.

MB_WARN_DATA_SPACE_FULL
If using a data space and the data space is out of storage, this warning is returned to remind the application program that no messages will be returned to this application program. This message will still be routed to CMIP services.

MB_WARN_EXIT_FAILURE
If using common storage area storage and the application program has indicated that it has had an unrecoverable error when returning to the read queue exit routine, this warning is returned to remind the application program that no messages will be returned to the application program. This message will still be routed to CMIP services.

MB_ERR_INVALID_LINK_ID
The value specified on the link identifier parameter does not refer to a valid connection.

MB_ERR_TRANSMIT
An apparent error occurred. Either there is a logic error in VTAM, or the MIBDisconnect function has been issued, but it has not completed.

MB_ERR_VTAM_INACTIVE
VTAM is inactive.

Example of function in an application program

The following example shows how the MIBSendDeleteRegistration function can be coded in an application program.

```
#include "acyaphdh.h"

int rc;
int LinkId;
MIBSendDeleteRegistration_t *MIBSendDeleteRegistration;

/*********************************************************/
/* Delete a registration for the object with local       */
/* identifier MyLocalId.                                 */
/*********************************************************/

rc = MIBSendDeleteRegistration(LinkId,
                               &InvokeId,
                               &MyLocalId,
                               NULL);
```

MIBSendRegister—MIB asynchronous registration function

Purpose

The MIB registration function must be called at least once in order for an application to access CMIP services or receive unsolicited messages. The MIB registration function can be called many times on any given MIB connection. For each call to the MIB registration function a unique local identifier must be provided by the caller. The local identifier can be used to distribute messages to the appropriate objects as they arrive over the connection. Because the local identifier must be provided on the registration call, it could be a pointer to a control block that could be directly referred to from the API header. The application program might also provide a handle for secondary routing. The size of the local identifiers is specified on the local identifier length parameter of the MIBConnect function.

A registered object can be a create handler for zero or more object classes. In other words, it can be responsible for handling CMIP create requests for instances of certain classes.

An application program specifies this property for an object by providing a list of classes on the call to MIBSendRegister when registering the object that is a create handler.

The responsibilities of a create handler are described in "Create handlers" on page 13, which describes create processing.

A registered object is an instance of one specific object class. However, it can act like an instance of other classes if appropriate. Allomorphism is the term used to describe an object which can act like an instance of more than one class. The usual reason for allomorphism is when an object acts like an instance of the classes of which its class is a subclass.

An application program specifies this property for an object by providing a list of classes on the call to MIBSendRegister when registering the object which acts allomorphic to other classes.

A response will be generated by CMIP services for each call to the CMIP services registration function. The invoke identifier field in the API header can be used to correlate the response to the initial registration request. The response can be of two possible types. If the registration was successful the response is of type MIB.RegisterAccept, otherwise the response is of type MIB.ServiceError.

The application program must correlate the response from CMIP services to the registration request, using the invoke identifier, and determine by the message type in the API header whether or not the registration completed successfully.

Declarations

The following declarations indicate the order of the parameters for this function.

```
typedef int MIBSendRegister_t(
              int,                /* link identifier - input       */
              unsigned int *,     /* returned invoke identifier -
                                     output                        */
              const void *,       /* local identifier - input      */
              const char *,       /* object class - input          */
              int,                /* name type - input             */
```

```
const char *,        /* distinguished name - input    */
const char *,        /* name binding object
                        identifier - input            */
unsigned int,        /* capability flags - input      */
unsigned int,        /* allomorphs count - input      */
char **,             /* allomorphs array - input      */
unsigned int,        /* create handlers count - input */
char **);            /* create handlers array - input */
```

Parameters

link identifier
Specifies the link identifier returned by the MIBConnect function.

returned invoke identifier
Specifies the invoke identifier. The invoke identifier is used to correlate this request with a response that arrives subsequently.

local identifier
Pointer to the local identifier of the object that is to be registered. Specify NULL for the local identifier parameter if only the distinguished name is provided.

object class
This parameter is the registered class of the object being registered. The class is required on all registration calls.

name type
This must be DN_OF_INSTANCE.

distinguished name
This is the distinguished name of the object instance being registered. To use the distinguished name in future calls to CMIP services, the &DN MIB variable can be used to refer to the distinguished name associated with the object instance (see "MIB variable format" on page 98).

name binding object identifier
This is the object identifier for the name binding to be used. If NULL is specified for this parameter, CMIP services chooses a name binding.

capability flags
A parameter used to specify special properties of the object being registered.

The value should be NO_CAPABILITIES if no special properties are desired or SUBTREE_MANAGER if the object being registered should be a manager of the subtree with its distinguished name as the root.

allomorphs count
This is the number of classes to which this object is allomorphic.

allomorphs array
This is an array of pointers to character strings, each of which is the object identifier of a class to which this object is allomorphic.

create handlers count
This is the number of classes for which this object is a create handler.

create handlers array
This is an array of pointers to character strings, each of which is the object identifier of a class for which this object is a create handler.

Return codes

0 The function was successful.

MB_ERR_ALLOC
An error occurred allocating storage. If MB_ERR_ALLOC is received by the application program from an API function and there is a corresponding REQS record in the VIT with a nonzero return code, the LPBUF pool is not large enough and should be increased.

MB_ERR_INVALID_CAPABILITY_FLAGS
The value specified for the capability flags parameter is not valid.

MB_ERR_CMIP_SERVICES_INACTIVE
CMIP services is inactive.

If using common storage area storage, the read queue exit routine stops functioning.

If using data space storage, messages are not put on the data space.

MB_ERR_DISTINGUISHED_MISSING
The distinguished name parameter was not provided.

MB_ERR_INVALID_LINK_ID
The value specified on the link identifier parameter does not refer to a valid connection.

MB_ERR_INVALID_INVOKE_ID
The invoke identifier parameter was not provided.

MB_ERR_LOCAL_ID_MISSING
A local identifier was not provided.

MB_ERR_MAX_OUTSTANDING
The value specified for the maximum outstanding requests parameter is not valid.

MB_ERR_NOT_REGISTERED
For common storage area storage, the application program has indicated that it has had an unrecoverable error when returning to the read queue exit routine or that the data space is out of storage. The registration will not be allowed.

MB_ERR_OBJECT_CLASS_MISSING
The object class name parameter was not provided.

MB_ERR_TRANSMIT
An apparent error occurred. Either there is a logic error in VTAM, or the MIBDisconnect function has been issued, but it has not completed.

MB_ERR_VTAM_INACTIVE
VTAM is inactive.

Example of function in an application program

The following example shows how the MIBSendRegister function can be coded in an application program.

```
char MyObjectName        ??(120??);
int LinkId;
int rc;
LocalId_t *MyObjectId;
unsigned int InvokeId;

rc = APIs.MIBSendRegister(LinkId,  /* This is the handle returned by
                                      MIBConnect.                  */
                   &InvokeId,      /* MIBSendRegister will store an
                                      invoke id, or correlator, for
```

```
                   the registration request here.*/
&MyObjectId,    /* This is the address of
                   the local id to be associated
                   with this object.            */
"1.3.18.0.0.2155", /* This is the object
                      class of this object.     */
DN_OF_INSTANCE, /* This parameter must
                   have this value.             */
MyObjectName,   /* This is the distinguished
                   name of this object.         */
NULL,           /* Use default name binding.    */
0,              /* no special capabilities      */
0,              /* no allomorphs                */
NULL,           /* no allomorphs                */
0,              /* not a create handler for any
                   class                        */
NULL);          /* not a create handler         */
```

MIBSendRequest—MIB queue request function

Purpose

Use this function when an application program needs to send VTAM-specific requests. For a list of these requests, refer to Chapter 10, "VTAM-specific requests and responses," on page 133.

Declarations

The following declarations indicate the order of the parameters for this function.

```
typedef int MIBSendRequest_t(
              int,                /* link identifier - input       */
              unsigned int *,     /* returned invoke identifier -
                                     output                        */
              const void *,       /* local identifier - input      */
              const char *);      /* message - input               */
```

Parameters

link identifier
Specifies the link identifier returned by the MIBConnect function.

returned invoke identifier
Specifies the invoke identifier. The invoke identifier is used to correlate this request with a response that arrives subsequently.

local identifier
Pointer to the local identifier of the object that is issuing the request.

message
This is a pointer to a formatted string which contains the string header and the request data.

Return codes

0 The function was successful.

MB_ERR_ALLOC
An error occurred allocating storage. If MB_ERR_ALLOC is received by the application program from an API function and there is a corresponding REQS record in the VIT with a nonzero return code, the LPBUF pool is not large enough and should be increased.

MB_ERR_CMIP_SERVICES_INACTIVE
CMIP services is inactive.

If using common storage area storage, the read queue exit routine stops functioning.

If using data space storage, messages are not put on the data space.

MB_WARN_DATA_SPACE_FULL
If using a data space and the data space is out of storage, this warning is returned to remind the application program that no messages will be returned to this application program. This message will still be routed to CMIP services.

MB_WARN_EXIT_FAILURE
If using common storage area storage and the application program has indicated that it has had an unrecoverable error when returning to the read queue exit routine, this warning is returned to remind the application

program that no messages will be returned to the application program. This message will still be routed to CMIP services.

MB_ERR_INVALID_INVOKE_ID
The invoke identifier parameter was not provided.

MB_ERR_LOCAL_ID_MISSING
A local identifier was not provided.

MB_ERR_INVALID_MAX_INVOKE_IDS
The value specified for the maximum outstanding requests parameter is not valid.

MB_ERR_MSG_MISSING
The message parameter was not provided.

MB_ERR_TRANSMIT
An apparent error occurred. Either there is a logic error in VTAM, or the MIBDisconnect function has been issued, but it has not completed.

MB_ERR_VTAM_INACTIVE
VTAM is inactive.

Example of function in an application program

The following example shows how the MIBSendRequest function can be coded in an application program.

```
int LinkId;
unsigned int InvokeId;
MIBSendRequest_t *MIBSendRequest;

/*********************************************************/
/* Retrieve information on the association with handle a1. */
/*********************************************************/

rc = MIBSendRequest(LinkId,
                    &InvokeId,
                    &MyLocalId,
                    "msg ACF.GetAssociationInfo("
                    "handle 'a1', info 11111111)");
```

MIBSendResponse—MIB queue response function

Purpose

Use this function when an application program needs to send a VTAM-specific response to CMIP services. This function is not used to send ROIVapdu, RORSapdu, or ROERapdu responses.

One message that is sent by MIBSendResponse is MIB.DeleteResponse. For a list of these responses, refer to Chapter 10, "VTAM-specific requests and responses," on page 133.

Declarations

The following declarations indicate the order of the parameters for this function.

```
typedef int MIBSendResponse_t(
               int,                 /* link identifier - input        */
               unsigned int,        /* invoke identifier - output     */
               const void *,        /* local identifier - input       */
               const char *,        /* source - input                 */
               const char *,        /* destination association
               const char *,           handle - input                 */
               const char *);       /* message - input                */
```

Parameters

link identifier
Specifies the link identifier returned by the MIBConnect function.

invoke identifier
This is the invoke identifier of the request which is being responded to with this API call.

local identifier
Pointer to the local identifier of the object that is responding. Specify the same identifier as the one specified in the request.

source
The distinguished name of the originator of the request. This can be used to override the source of the message. This is used to resolve any appearance of the MIB variable distinguished name. Specify NULL if you do not choose to specify a value.

destination association handle
This is the association identifier of the association that is to be used to send the response. It is required and must be the same as the association handle that was received on the message that is being answered.

message
This is a pointer to a formatted string which contains the string header and the response data.

Return codes

0 The function was successful.

MB_ERR_ALLOC
An error occurred allocating storage. If MB_ERR_ALLOC is received by the application program from an API function and there is a corresponding REQS record in the VIT with a nonzero return code, the LPBUF pool is not large enough and should be increased.

MB_ERR_ASSOC_HANDLE_MISSING
The association handle parameter was not provided.

MB_ERR_CMIP_SERVICES_INACTIVE
CMIP services is inactive.

If using common storage area storage, the read queue exit routine stops functioning.

If using data space storage, messages are not put on the data space.

MB_WARN_DATA_SPACE_FULL
If using a data space and the data space is out of storage, this warning is returned to remind the application program that no messages will be returned to this application program. This message will still be routed to CMIP services.

MB_WARN_EXIT_FAILURE
If using common storage area storage and the application program has indicated that it has had an unrecoverable error when returning to the read queue exit routine, this warning is returned to remind the application program that no messages will be returned to the application program. This message will still be routed to CMIP services.

MB_ERR_INVALID_LINK_ID
The value specified on the link identifier parameter does not refer to a valid connection.

MB_ERR_INVALID_INVOKE_ID
The invoke identifier parameter was not provided.

MB_ERR_LOCAL_ID_MISSING
A local identifier was not provided.

MB_ERR_MSG_MISSING
The message parameter was not provided.

MB_ERR_TRANSMIT
An apparent error occurred. Either there is a logic error in VTAM, or the MIBDisconnect function has been issued, but it has not completed.

MB_ERR_VTAM_INACTIVE
VTAM is inactive.

Example of function in an application program

The following example shows how the MIBSendResponse function can be coded in an application program.

```
const char *AssocHandle;
int LinkId;
int rc;
void *LocalId;
unsigned int InvokeId;

rc = MIBSendResponse(LinkId,InvokeId,
                     LocalId,NULL,AssocHandle,,
                     "MIB.DeleteResponse(1,processingFailure)");
```

Chapter 5. Read queue exit routine

For the common storage area (CSA) interface, the read queue exit routine is entered when VTAM CMIP services needs to notify or send data to the application program.

For the data space interface, the read queue exit routine is entered when VTAM CMIP services needs to notify the application program that:
- There are messages on the data space to be read
- CMIP services is terminating
- The data space is full

The requirements for callers of the read queue exit routine are:

Location
User private

Key Same key that was used when the MIBConnect function was called

State Supervisor state

AMODE
31-bit

Residency mode
Any

ASC mode
Primary

Interrupt status
Enabled

Dispatchable unit mode
TCB

Locks No locks held

ENQs No ENQs held

@space
Same address space from which MIBConnect was issued

The data passed to the read queue exit routine is located in CSA storage and is allocated in the same key that was used when the MIBConnect function was called. The data is not fetch protected, so any key can be used to copy it. The read queue exit routine should not attempt to free any storage passed to it. Storage is freed automatically when the exit routine terminates. Application programs can vary depending on product data and queuing structures. The following list gives recommendations for the read queue exit routine:
- Use the contents of the user data field located in register 6 to set up the environment. This field can be the address of an autodata area to improve performance, or it can be NULL.
- Save the calling application program's registers in the provided save area.
- Check the VTAM reason codes to determine why the exit routine was called and what action should be taken. For a list of reason codes, refer to "VTAM reason codes (for data space)" on page 89 and "VTAM reason codes (for CSA)" on page 88.

Read queue exit routine for the CSA interface

This section describes how the read queue exit routine is called for the application program when CSA storage is used for receiving data from CMIP services.

VTAM reason codes (for CSA)

Reason code
Explanation

0 Data is being passed to the read queue exit routine.

MB_ERR_CMIP_SERVICES_INACTIVE
CMIP services has terminated. Signal the application program main task to issue the MIBDisconnect function. No data is passed for this reason code.

Note: Your read queue exit routine should be coded to ignore unrecognized reason codes and set the return code to 0.

For a reason code of zero, copy any data presented from CSA storage to private storage. Then queue the copied data to the appropriate task. CMIP services examines the return code only if the read queue exit routine is driven with a reason code of 0. Set register 15 as follows:

Return code
Explanation

0 The read queue exit routine was successful.

8 The read queue exit routine had a temporary internal processing failure; for example, it is out of storage.

CMIP services builds an ROER if the message passed to the exit is a confirmed request of type ROIVapdu. The read queue exit routine continues to function.

16 The read queue exit routine had a permanent internal processing failure.

CMIP services builds an ROER if the message passed to the exit routine is a confirmed request of type ROIVapdu. It also builds these ROERs for any subsequent confirmed ROIV requests until the application program disconnects from the API. The read queue exit routine does not continue to receive data. It is driven again only if CMIP services terminates of if application program calls the MIBDisconnect function and then calls the MIBConnect function again.

Registers upon entry (for CSA)

The following list shows the registers upon entering the read queue exit routine.

Register
Contents

1 Address of variable length parameter list. The end of the parameter list is indicated by the number 1 in the high-order bit of the last word. For details about the parameter list, refer to “Parameter list (for CSA)” on page 89.

6 Contents of the user data field that was passed on the MIBConnect function.

13 Address of an 18 fullword save area.

14 Return address.

15 Entry point address of the exit routine.

Registers upon termination (for CSA)

The following list shows the registers upon terminating the read queue exit routine.

Register
Contents

0-14 Unchanged, restored to values on entry.

15 Return code:

- **0** Successful; input data processed.
- **8** Unsuccessful; storage failure.
- **16** Unsuccessful; terminate the exit routine.

Parameter list (for CSA)

The following list shows the parameter list for the read queue exit routine. The decimal value is first, followed by the hexadecimal value in parentheses.

Offset Description

0 (0) VTAM reason code. For a list, refer to "VTAM reason codes (for CSA)" on page 88.

4 (4) Address of API header.

8 (8) Address of string header. Refer to "Description and example of the string" on page 48 for details.

12 (C) Length of API header + string header + CMIP string. Four-byte field that represents the length of the total data to be copied.

Read queue exit routine for data space storage

This section describes how the read queue exit routine is called for the application program when data space storage is used for receiving data from CMIP services.

VTAM reason codes (for data space)

Reason code
Explanation

MB_DATA_ON_DATA_SPACE
CMIP services has placed one or more messages in the data space.

MB_WARN_DATA_SPACE_FULL
Data space storage is full. Signal the appropriate application task to issue the MIBDisconnect function.

MB_ERR_CMIP_SERVICES_INACTIVE
CMIP services has terminated. Signal the appropriate application task to issue the MIBDisconnect function.

Note: Your read queue exit routine should be coded to ignore unrecognized reason codes and set the return code to 0.

Contents of register 15 are not examined when read queue exit routine returns. Any messages in the data space are the responsibility of the application program. CMIP services does not perform any special processing to build ROERs for these messages. The read queue exit routine continues to be driven every time the number of waiting messages in the data space goes from zero to one until the application program disconnects from the API.

Registers upon entry (for data space)

The following list shows the registers upon entering the read queue exit routine.

Register
Contents

1 Address of variable length parameter list. The end of the parameter list is indicated by the number 1 in the high-order bit of the last word. For details about the parameter list, refer to "Parameter list (for data space)."

6 Contents of user data field which was passed on the MIBConnect function.

13 Address of an 18 fullword save area.

14 Return address.

15 Entry point address of the exit.

Registers upon termination (for data space)

The following list shows the registers upon terminating the read queue exit routine.

Register
Contents

0-14 Unchanged, restored to values on entry

15 Zero

Parameter list (for data space)

The following list shows the parameter list for the read queue exit routine. The decimal value is first, followed by the hexadecimal value in parentheses.

Offset Description

0 (0) VTAM reason code. For a list, refer to "VTAM reason codes (for data space)" on page 89.

Chapter 6. Dequeue and release routines for data space storage

The dequeue routine retrieves messages from the data space, one at a time.

The release routine frees the data space storage for each message that has been processed.

The release and dequeue routines are non-reentrant per application program.

This chapter describes:
- Format of data on the data space
- Dequeuing a buffer with the dequeue routine
- Releasing a buffer with the release routine

Format of data on data space

The format for data on data space storage is shown in the following list. The decimal value is first, followed by the hexadecimal value in parentheses.

Offset Description

0 (0) Address of API header (within data space). Refer to the declaration of APIhdr in ACYAPHDH under Appendix A, "C language header file (ACYAPHDH)," on page 229.

4 (4) Address of string header (within data space). Refer to "Description and example of the string" on page 48 for details.

8 (8) Length of API header + string header + CMIP string. This is a 4-byte field that represents the length of the total data to be copied.

The requirements for callers of the read queue exit routine are:

Location
User private

Key Key 6

State Supervisor state

AMODE
31-bit

Residency mode
Any

ASC mode
Access Register mode

Interrupt status
Enabled

Dispatchable unit mode
TCB

Locks No locks held

ENQs No ENQs held

@space
User address space

Dequeueing a buffer with the dequeue routine

When the application program is notified by the read queue exit routine that data is on the data space (MB_DATA_ON_DATA_SPACE), the application program must call the dequeue routine to receive the data. The dequeue routine dequeues the buffer until register 0 returns a 0 buffer count.

The dequeue routine address is returned on the MIBConnect function in the interface control block. For information about the data space vector parameter, refer to page 62.

Input to the dequeue routine

This routine is serially reusable per queue. If the application program attempts to overlap execution of this routine, the results are unpredictable.

General registers
Explanation

0 Value of the field RIV10NMI in the ISTRIV10_t structure filled in by MIBConnect.

1 Unused

2-13 Undefined

14 Return address

15 Entry point address

Access registers
Explanation

0 Undefined

1 ALET for interface data space

2-15 Zero

Output for dequeue routine

General registers
Explanation

0 Count of remaining buffers

1 Address of buffer that is in the data space or zero if no buffer exists

2-13 Restored to input values

14 Return address

15 Return code:

- **0** Buffer is dequeued. The address is in register 1.
- **8** No buffers available.
- **16** VTAM is terminating. The application program's TPEND exit routine is driven. Do not continue calling the interface routines. Cease all reference to interface control blocks.

Access registers

	Explanation
0	Undefined
1	ALET for interface data space
2-14	Restored to input value
15	Undefined

Releasing a buffer with the release routine

To release a previously dequeued buffer, the application program must call the release routine. The release routine address is returned on the MIBConnect function in the interface control block. For information about the data space vector parameter, refer to page 62.

This module is serially reusable per queue. If the application program attempts to overlap execution of this module, the results are unpredictable.

Input to the release routine

General registers

	Explanation
0	Value of the field RIV10NMI in the ISTRIV10_t structure filled in by MIBConnect.
1	Address of buffer to be released
2-13	Undefined
14	Return address
15	Entry point address

Access registers

	Explanation
0	Undefined
1	ALET for interface data space
2-15	Zero

Output to the release routine

General registers

	Explanation
0-1	Undefined
2-13	Restored to input values
14	Return address
15	Return code:
	0 Buffer released.
	16 VTAM is terminating. The application program's TPEND exit routine is driven. Do not continue calling the interface routines. Cease all reference to interface control blocks.

Access registers

	Explanation
0	Undefined
1	ALET for interface data space
2-14	Restored to input value
15	Undefined

Abnormal exits

If the buffer being released is either not allocated or is incorrect, the results are unpredictable.

Chapter 7. Rules for constructing standard CMIP strings

This section describes how to look at the ASN.1 source files and read the syntax to enable you to build a string that can be sent to CMIP services. Almost all of the data types supported by ASN.1 are supported by VTAM CMIP services. VTAM CMIP services does not support the following data types:

- GraphicString (except for the default character set, which is supported)
- TeletexString and VideotexString
- EXTERNAL data type
- Contained subtypes
- Inner subtyping
- Real value
- Constructed value
- Named bit strings

Overview

CMIP services includes a management information base (MIB) application program interface (API), which application programs use to send information to CMIP services. Application programs send data to CMIP services by using API functions, which are described under Chapter 3, "Overview of CMIP services API functions," on page 41. The data sent in some of the parameters of the API functions can be in any format that is accepted as standard ASN.1 syntax. ASN.1 syntax is the data definition language used by OSI management.

This section describes how application programs can send data to CMIP services (using the API functions) and how CMIP services sends data to application programs.

The application program can send strings that are composed of values that are specified according to the rules in the ASN.1 syntax. For a particular ASN.1 syntax, an application program has some flexibility in the exact format of a string.

CMIP services returns information to application programs in a specific format. For example, when the application program sends a string to CMIP Services that includes a BOOLEAN value, the application program can use a variety of formats. But when CMIP services sends a BOOLEAN value in a string to the application program, CMIP services uses only one format for BOOLEAN values.

How application programs format data to be sent to CMIP services

When calling the MIBSendRequest or MIBSendResponse functions, the application program provides a zero-terminated string that includes the following:

- The word **msg**
- A blank
- The name of an ASN.1 module
- A period
- The name of a type within that ASN.1 module
- Values for all of the fields associated with that type

For example, the following zero-terminated string could be passed as the fourth parameter to the MIBSendRequest function.

```
"msg ACF.Release (a1)"
```

When calling the MIBSendCmipRequest or MIBSendCmipResponse functions, the application program provides a zero-terminated string that includes only the values for all of the fields associated with the type listed in the ANY DEFINED BY table for the specified operation-value or error-value.

For example, to send a GET request by the MIBSendCmipRequest function, the second parameter of the MIBSendCmipRequest function should be three (operation-value for GET) and the third parameter of MIBSendCmipRequest function could be the following zero-terminated string:

```
"(baseManagedObjectClass 2.9.3.2.3.13,"
" baseManagedObjectInstance "
"   (distinguishedName "
"      '1.3.18.0.2.4.6=NETA;2.9.3.2.7.4=(name GEORGE)'),"
" attributeIdList (2.9.3.2.7.35,2.9.3.2.7.5))"
```

Each value is made up of a <label> <value> pair. The <label> is the identifier that appears in ASN.1 NamedTypes. See clause 12.5 of ISO-8825 for the formal definition of a NamedType.

In the following example, a, b, and c are possible labels. For the field with data type D, the type name is used as a label. Using the type name as a label is necessary only when the ASN.1 syntax was defined without labels for all SET and SEQUENCE fields. If the type name is used for a data type that has a label, the type name is rejected.

```
A ::= SEQUENCE
      {
          a INTEGER,
          b OBJECT IDENTIFIER,
          c C,
            D
      }
```

Labels can always be specified, but they are required only to resolve ambiguity in the ASN.1 definition. Because it is difficult to know when ambiguity exists, use the following rules when building strings to send to CMIP services:

- Labels are required on members of a SET construct, because the members of the SET can be specified in any order.
- A label is required to resolve a CHOICE; otherwise CMIP services cannot determine which choice was selected by the application program.
- It is recommended that members of a SEQUENCE be identified with a label. Labels are required only in situations where an optional member is intentionally omitted and subsequent members follow. However, unless every member of a sequence is specified, or the optional members that are intentionally omitted are located at the end of the SEQUENCE, it is simpler to identify all members with a label.
- Elements of a SET and SEQUENCE and the element of a CHOICE are surrounded by parentheses.

The <value> portion of the <label> <value> pair can be specified in the following ways:

- *Primitive* data types, such as BOOLEAN and INTEGER, that are not composed of one or more instances of other data types

- *Constructed* data types, such as SEQUENCE and SET
- *Hexadecimal* basic encoding rules (BER), which can be used for all ASN.1 types except CHOICE and ANY DEFINED BY.

Any of the five following formats are recognized by CMIP services, but CMIP services always returns explicit value notation if there are no insurmountable errors encountered during the decoding of incoming strings. If errors are encountered, the hexadecimal BER format explained in "Hexadecimal BER format" on page 100 is used.

Explicit value format

In the explicit value format, the actual value of the primitive data type is given. For example, an application program can specify 1234 as the value of an INTEGER data type. Each of the primitive data types has a unique explicit value notation and these are explained in "Primitive ASN.1 data types" on page 101. Examples are TRUE and FALSE for BOOLEAN types, -3.125 for REAL types, and 1001001 for BIT STRINGS.

Values can be formatted and sent to the API enclosed in single or double quotation marks. Quotation marks are required if the value contains a space. Use the same kind of quotation mark to begin and end the value. The quotation marks are ignored by CMIP services.

ASN.1 value format

The value format is based on an ASN.1 module, as shown in the following example.

```
    A ::= INTEGER

    a INTEGER ::= 1
    b INTEGER ::= 2
    c INTEGER ::= 3
    d A       ::= 4

    B ::= SET {
       f   [1] INTEGER,
       g   [2] INTEGER,
       h   [3] INTEGER
       }
    C ::= SEQUENCE {
       f   [1] INTEGER,
       g   [2] INTEGER,
       h   [3] INTEGER
       }

Values for A may be specified as:
        a
        b
        d
        12
Values for B may be specified as:
        (f a, h d, g 34)
        (h 138, f d, g 34)
Values for C may be specified as:
        (c, 12, 19)
        (f c, 12, h 19)
        (f c, g 12, h 19)
```

The application program can specify a, b, c, or d as the <value> portion of a <label> <value> pair. If the value appears in a context that might be ambiguous,

such as for the value of the g field in the SET B, the appropriate <label> must accompany the <value>. The labels can be omitted when specifying values for C, because there is no ambiguity. The labels can never be omitted when specifying values for B, because A is optional and without a label for B, it is not possible to determine whether the value is for A or B.

CMIP services, using information from the compiled ASN.1 modules, verifies that the value and type are compatible.

MIB variable format

MIB variables are values that can be set in CMIP services by an object, and then referred to later in a string. These values can be specified as MIB variables by the application program in any string. CMIP services substitutes the actual values.

MIB variables are denoted with an ampersand (&) as the first character of the variable name. The API checks to make sure that the type of the MIB variable and the type of the type reference are compatible.

CMIP services includes a set of predefined MIB variables that can be used in any string, by any object:

&DN Represents the distinguished name of the originator of a string that is passed to the API. The API uses its knowledge of the source of the string to provide the appropriate distinguished name. The name can be used by an object that is registered with CMIP services to identify itself when it sends a string. The API supplies the distinguished name that corresponds to the local identifier provided on the request.

&IID Represents the invoke identifier of the current string. This can be used in a response or when initiating a request. On requests, this MIB variable allows the sender of a string to build the string without knowing the invoke identifier. For all requests, the invoke identifier is not required because the MIB functions assign the invoke identifier after they receive the string. Therefore the API can fill in the value for the invoke identifier once it has been assigned.

&GTM
Represents the current time in Generalized time format. For more information, refer to "Time types" on page 112.

```
yyyy/mm/dd-hh:mm:ss.0
```

&UTM
Represents the current time in Universal time format. For more information, refer to "Time types" on page 112.

```
yyyy/mm/dd-hh:mm:ss.0
```

&OC Represents the managed object class of the originator of the string. This variable allows the application program to use generic strings in responding to requests, without having to customize them for each object class it supports. In any response from an unregistered object or when allomorphism is being exercised, this variable cannot be used.

The following example shows how to use these MIB variables:

```
Arg =
  "(managedObjectClass &OC,                                    "
  " managedObjectInstance (distinguishedName &DN),             "
  " currentTime &GTM,                                          "
  " attributeList ((attributeId 2.9.3.2.7.5,                   "
```

```
  "                (distinguishedName                          "
  "                 (((attributeType 1.3.18.0.2.4.6,           "
  "                    attributeValue MYNETID),                "
  "                   (attributeType 2.9.3.2.7.4,              "
  "                    attributeValue (name \"MYCPNAME\")))))," 
  "               (attributeId 2.9.3.2.7.35, enabled)          "
  "              )                                             "
  ")                                                          ";

rc = MIBSendCmipResponse(LinkId,
                         OldInvokeId,
                         1,          /* last in chain */
                         1,          /* success       */
                         3,          /* GET response  */
                         Arg,
                         LocalId_ptr,
                         NULL,
                         OldAssocHandle,
                         NULL);
```

CMIP Services substitutes the appropriate values for the variables &OC, &DN, and &GTM.

Note: The many extra spaces in the response string will be ignored by CMIP services, though they will lead to extra processing overhead.

Constructed value format

The constructed types, SET, SEQUENCE, SET OF, and SEQUENCE OF and the CHOICE types use constructed value format. In this format, the value of a <label> <value> pair is surrounded by parentheses and contains other <label> <value> pair specifications separated by commas, as is shown in the following ASN.1 definition:

```
A ::= SEQUENCE
      {
          a INTEGER,
          b BIT STRING,
          c BOOLEAN
      }
```

The invoking application program specifies the following across the API:

```
(a 12, b 11011011, c TRUE)
```

To nest constructed data types, use multiple sets of parentheses. Note that the number of parentheses does not correspond directly to the number of braces in the ASN.1. It corresponds to the number of constructed data types that occur. For example, an application program could specify

```
(a 12, b (1, 2, 3, 4), c TRUE, d (111, 1101110, 11000))
```

to be sent to the API to correspond to the following ASN.1 definition:

```
A ::= SEQUENCE
      {
          a [0] INTEGER DEFAULT 0,
          b [1] SEQUENCE OF INTEGER,
          c [2] BOOLEAN OPTIONAL,
          d [3] B
      }
B ::= SEQUENCE OF C
C ::= BIT STRING
```

The numbers specified in square brackets in the ASN.1 of the previous example refer to the tagging that is used when exchanging strings between systems. Because the identifier of the named type (in this case, `a, b, c,` or `d`) corresponds not only to the type reference but also to the tagging, it is not necessary to specify the tagging across the API. Tags are determined automatically by CMIP services.

The words DEFAULT and OPTIONAL in an ASN.1 definition indicate that those fields can be omitted in an instance of type reference A. DEFAULT means the field *a* can be omitted. If it is omitted, CMIP services interprets the field as having a value the default value specified in the syntax. In the previous example, zero is assigned to the field with label A. If it is not omitted, CMIP services does not assign DEFAULT fields default values. Application programs that receive strings containing DEFAULT fields must be able to understand and interpret the omission of the field.

OPTIONAL means that the field *c* does not have to be specified. When it is not specified, CMIP services does not interpret the field.

Hexadecimal BER format

Hexadecimal BER format is the hexadecimal value contained in the BER, enclosed in angle brackets. Hexadecimal BER format is the final format that can be used to specify a value. In some cases when CMIP services cannot decode a string sent by another CMIP services, CMIP services sends the string to the application program in hexadecimal BER format.

In this format, the value is enclosed in less than (<) and greater than (>) symbols, and consists of zero or more hexadecimal digits. In all but one case, the hexadecimal digits represent the BER encoding of the <value> portion of a particular field. For example, the value of a BOOLEAN in BER is specified as a single **octet**, with nonzero values representing true. An octet is a byte. To specify a true value for field fred to the API in hexadecimal BER format, the application program specifies:

```
... fred <01> ...
```

When specifying a value for an ANY type, the application program is required to specify the entire BER field, including the tag, length, and value portions. It cannot specify only the value, because the ANY type cannot understand what the possible types are. For example, if the same application program specifies a BOOLEAN value of true to the API for a field called fred that is an ANY type, the following should be specified:

```
... fred <010101> ...
```

In this example, the first octet represents the tag, which is a universal tag for BOOLEAN. For a full description of how BER tags are encoded see the BER standard.

The second octet represents the length of the value portion, a length of 1 octet, and the value is as specified previously.

An application program should not use hexadecimal BER when sending information to the API, because error and subtype checking that is normally performed by the API code is not applied to the BER value. The value is assumed to be correctly formed and is inserted into the BER buffer at the appropriate location.

Another potential problem is the use of the hexadecimal BER format for ANY types, because improper tags and lengths can be introduced. Hexadecimal BER format is necessary when an INTEGER that is longer than four octets needs to be shipped. Hexadecimal BER allows the application program to circumvent any limitations imposed by the API, but you might encounter problems.

When CMIP services receives strings from an application program, CMIP services attempts to decode the strings into a combination of explicit values and constructed values. During decoding, if CMIP services encounters an error in a primitive data type, CMIP services sends to the application program the value for that type in hexadecimal BER format. For example, if the contents of an INTEGER field are too large to fit within four octets, CMIP services sends the application program the INTEGER value in hexadecimal BER format.

If CMIP services encounters an error in a constructed data type or a decision data type, such as ANY DEFINED BY or CHOICE, CMIP services sends the application program the entire contents of the constructed or decision type in a single hexadecimal BER value. For example, if CMIP services does not recognize the value of an OBJECT IDENTIFIER, the OBJECT IDENTIFIER value is sent to the application program in hexadecimal BER format.

Primitive ASN.1 data types

Primitive types within ASN.1 are those types that are not constructed or cannot be broken down into more primitive types. They correspond to the normal data types encountered in many programming and data definition languages.

The term *primitive type* should not be confused with *primitive encoding* as defined in the BER standard. Some primitive types, such as BIT STRINGs, can actually be encoded in a constructed manner. However, in this case, all of the components must be of the same type as the constructed BIT STRING.

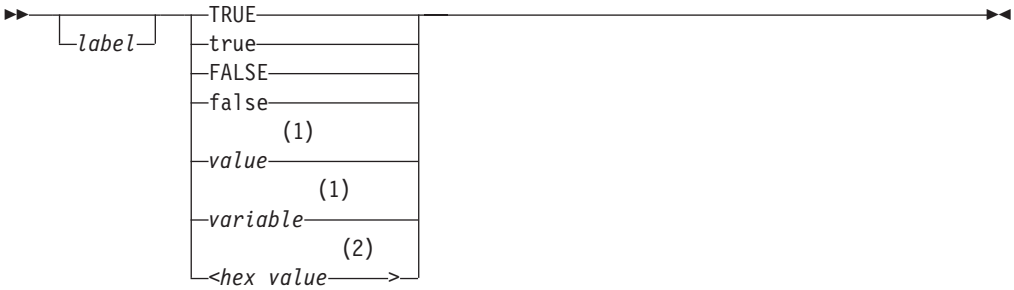
The following sections describe:
- How an application program sends the type to CMIP services
- How CMIP services sends the type to an application program

BOOLEAN type

BOOLEAN types can have one of two values: true or false.

How an application program sends a BOOLEAN value to CMIP services

An application program can send a BOOLEAN value to CMIP services in any of the following forms:



Notes:

1 Values and variables specified in this position must resolve to a BOOLEAN value.

2 When specifying a value in this format, be aware that the BER representation consists of a single octet, with X'00' representing false, and any other value representing true.

An application program can specify a BOOLEAN value as shown in the following examples:

```
TRUE
FALSE
true
false
```

How CMIP services sends a BOOLEAN value to an application program

CMIP services sends one of the following BOOLEAN values:

- TRUE
- FALSE

CMIP services places labels in the string for all elements of the syntax that are present.

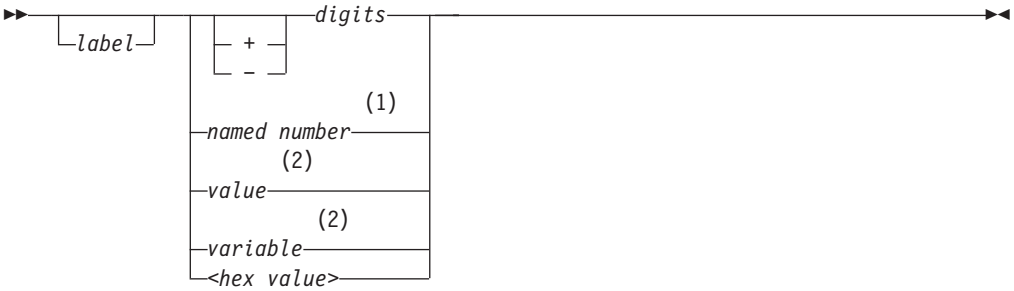
If CMIP services cannot decode the value, CMIP services sends the value to the application program in hexadecimal BER format, enclosed in delimiters. See "Hexadecimal BER format" on page 100 for a description.

INTEGER type

INTEGER types represent integer numbers. An INTEGER value can be either positive or negative. INTEGER values are expressed as the explicit value of the integer, which is the actual value of the integer. For example, an application program can specify 1234 as an INTEGER value. The minimum value is -2147483648; the maximum value is 2147483648.

How an application program sends an INTEGER value to CMIP services

An application program can send an INTEGER value to CMIP services in any of the following forms:



Notes:

1 The ASN.1 compiler recently introduced support for named numbers, and this support is expected to be added to the API in the very near future. When it is, the API will output named integer values by giving the value identifier.

2 Values and variables specified in this position must resolve to an INTEGER value.

The following example shows how the ASN.1 syntax might define an INTEGER value.

```
X ::= INTEGER
SlowModemSpeed ::= INTEGER {
        slowest  (300),
        slower   (1200),
        slow     (2400)
        }

A value for X would be:
      123
Values for SlowModemSpeed would be:
      300
      2400
```

How CMIP services sends an INTEGER value to an application program

CMIP services sends INTEGER values as strings of decimal digits, possibly preceded by a minus sign (–). INTEGER values are always represented by their numeric values.

CMIP services places labels in the string for all elements of the syntax that are present.

If CMIP services cannot decode the value, CMIP services sends the value to the application program in hexadecimal BER format, enclosed in delimiters. See "Hexadecimal BER format" on page 100 for a description. For example, if CMIP services encounters an INTEGER value longer than four octets, CMIP services sends the value to the application program in hexadecimal BER format.

ENUMERATED type

The values for ENUMERATED types are expressed as explicit values that are symbolic, rather than numeric.

How an application program sends an ENUMERATED value to CMIP services

An ENUMERATED can be formatted and sent to CMIP services in the following forms:

Notes:

1 Values and variables specified in this position must resolve to an ENUMERATED value.

The following example shows how the ASN.1 syntax might define an ENUMERATED value.

```
X ::= ENUMERATED  {
     val1 (0),
     val2 (1),
     val3 (2)
     }

Values for X would be:
      val1
      val3
```

How CMIP services sends an ENUMERATED value to an application program

CMIP services sends an ENUMERATED value as a symbolic ASCII string that corresponds to the value found in the BER.

ENUMERATED values are always represented by the name of the value, not the corresponding integer value.

CMIP services places labels in the string for all elements of the syntax that are present.

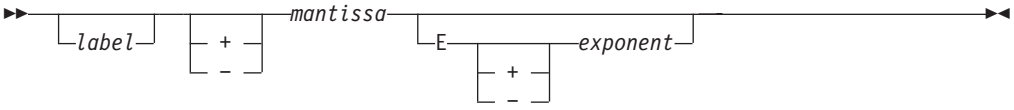
If CMIP services cannot decode the value, CMIP services sends the value to the application program in hexadecimal BER format, enclosed in delimiters. See "Hexadecimal BER format" on page 100 for a description.

REAL type

REAL types represent real values.

How an application program sends a REAL value to CMIP services

An application program can send a REAL value to CMIP services in any of the following forms:



The application program is required to use the hexadecimal BER format for specifying REAL values.

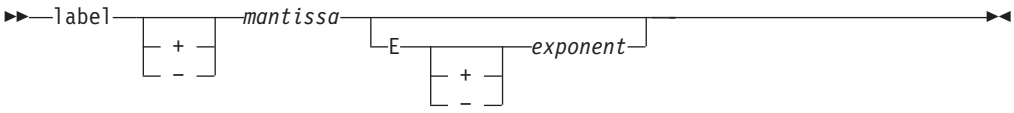
The following example shows how the ASN.1 syntax might define a REAL value.

```
X ::= REAL

Values for X would be:
      "3.14"
      "0.0"
      "-14.33e-05"
```

How CMIP services sends a REAL value to an application program

CMIP services sends REAL values to an application program in the following format:



CMIP services places labels in the string for all elements of the syntax that are present.

Under the OS/2® operating system, CMIP services sends REAL values according to the criteria used for the output of %lg in *printf()*. CMIP services sends the smallest number of characters that can be used to represent the number.

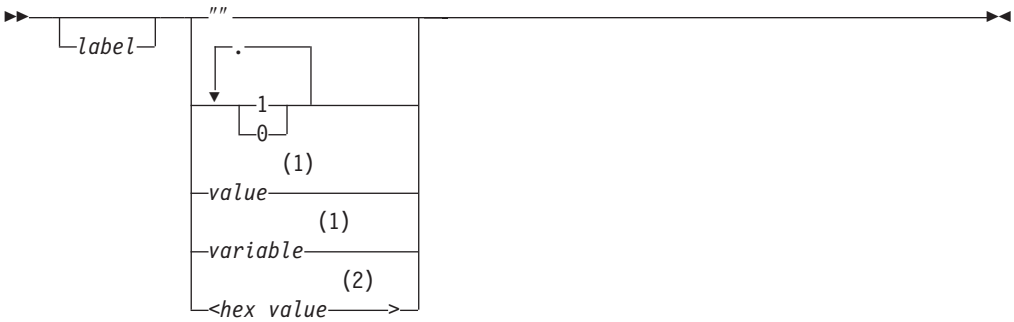
If CMIP services cannot decode the value or if CMIP services exists on an operating system other than OS/2, CMIP services sends the value to the application program in hexadecimal BER format, enclosed in delimiters. See "Hexadecimal BER format" on page 100 for a description.

BIT STRING type

The BIT STRING type represents a string of bits. There is no limit to the length of the string.

How an application program sends a BIT STRING to CMIP services

An application program can send a BIT STRING to CMIP services in any of the following forms:



Notes:

1 Variables specified in this position must resolve to a BIT STRING.

2 When specifying a value in this format, remember that the BER representation of a BIT STRING always begins with an octet that signifies the number of unused bits in the final octet of the value. Omitting this extra octet results in decoding errors by the receiver.

The bit strings are sent to CMIP services as part of a character string, using the characters B'1' and B'0' to represent on and off. The application program can also specify a *null* BIT STRING by entering two quotation marks, either single ('') or double (""). A null BIT STRING has a length of zero.

How an application program specifies a BIT STRING value

The following example shows how the ASN.1 syntax might define a BIT STRING.

```
X ::= BIT STRING {
     val1 (0),
     val2 (1),
     val3 (2)
     }

Values for X would be:
      001  - means val3 is turned on, the others are off
      100  - means val1 is turned on, the others are off
      111  - val1, val2, val3 are all on
```

How CMIP services sends a BIT STRING to an application program

CMIP services sends BIT STRINGs as strings of digits, without enclosing them in quotation marks. When CMIP services receives a null BIT STRING from an application program, CMIP services sends the null BIT STRING as two double quotation marks.

CMIP services places labels in the string for all elements of the syntax that are present.

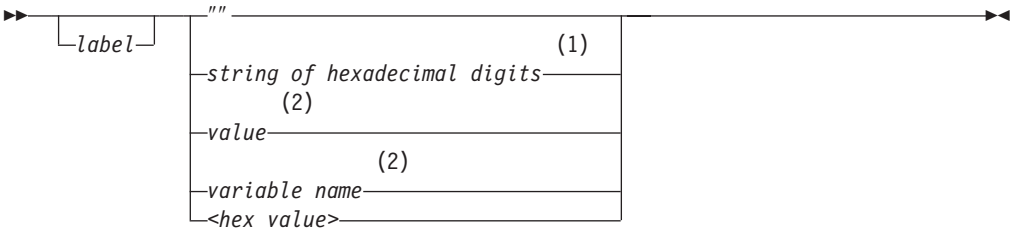
If CMIP services cannot decode the value for a BIT STRING, including null BIT STRINGS, CMIP services sends the value to the application program in hexadecimal BER format, enclosed in delimiters. See "Hexadecimal BER format" on page 100 for a description.

OCTET STRING type

The OCTET STRING type represents a string of hexadecimal digits.

How an application program sends an OCTET STRING to CMIP services

An application program can send an OCTET STRING to CMIP services in any of the following forms:



Notes:

1 The formatted string to be sent to CMIP services must have an even number of hexadecimal digits.

2 Variables specified in this position must resolve to OCTET STRINGs.

An application program can send OCTET STRINGs to CMIP services as strings of an even number of hexadecimal digits, using the character representation of the hexadecimal digits '0' through '9' and 'A' through 'F'. Both uppercase and lowercase letters can be used. When CMIP services returns the OCTET STRING, CMIP services uses uppercase letters.

Application programs can have OCTET STRINGs that have a length of zero. Such OCTET STRINGS are *null* OCTET STRINGs. For null OCTET STRINGs, the

application program should format the string with two quotation marks with no intervening characters. The application program can specify either single or double quotation marks.

An application program can also specify OCTET STRINGs as hexadecimal BER, although this format is essentially the same as the explicit value format, with different delimiters.

How an application program specifies an OCTET STRING

The following example shows how the ASN.1 syntax might define an OCTET STRING.

```
X ::= OCTET STRING (SIZE(2))

If X has a hexadecimal value of 01AB, the string passed to or from the
API is:
      F0F1C1C2
```

How CMIP services sends an OCTET STRING to an application program

CMIP services sends an OCTET STRING in explicit value format. CMIP services sends a null OCTET STRING as two double quotation marks when it sends the string.

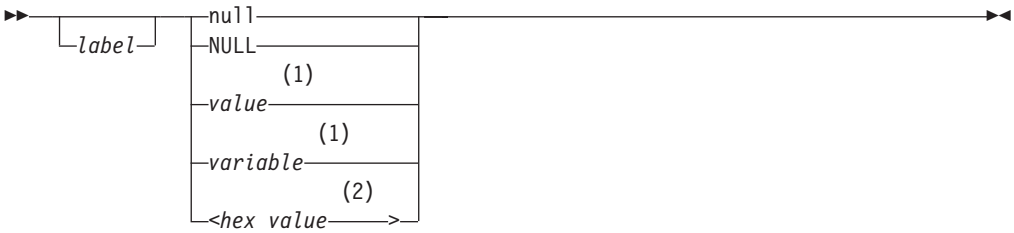
CMIP services places labels in the string for all elements of the syntax that are present.

NULL type

A NULL type is used for optional input parameters for which the application program does not specify a value.

How an application program sends a NULL value to CMIP services

An application program can send a NULL value to CMIP services in any of the following forms:



Notes:

1 Variables specified in this position must resolve to NULL.

2 When specifying a value in this format, remember that the BER representation of a NULL consists only of a tag and length field that indicates tha the length is zero. Therefore, the proper representation of hexadecimal BER should be “<>”.

An application program can specify a NULL value by specifying:
- The character string NULL
- An ASN.1 value label that resolves to a NULL value
- A MIB variable that resolves to a NULL value

How an application program specifies a NULL value

The following example shows how the ASN.1 syntax might define a BIT STRING value.

```
X ::= NULL

The value for X is:
      NULL
```

How CMIP services sends a NULL value to an application program

CMIP services sends a NULL value as the uppercase string `NULL`.

CMIP services places labels in the string for all elements of the syntax that are present.

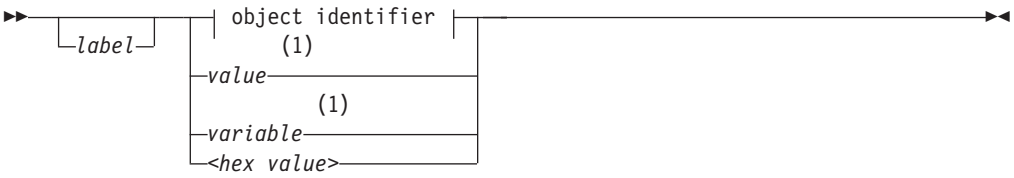
If CMIP services cannot decode the value, CMIP services sends the value to the application program in hexadecimal BER format, enclosed in delimiters. See "Hexadecimal BER format" on page 100 for a description.

OBJECT IDENTIFIER type

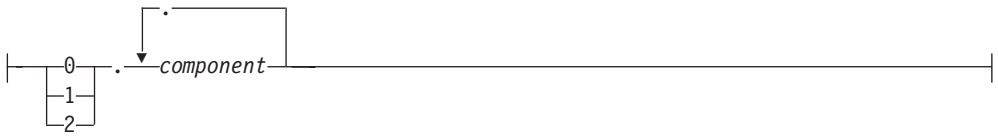
OBJECT IDENTIFIERs (OIs) serve within OSI management as universally unique codepoints to represent object classes, specific values, or identities of registered parts of an object class.

How an application program sends an OBJECT IDENTIFIER to CMIP services

An application program can send an OBJECT IDENTIFIER to CMIP services in any of the following forms:



object identifier:



Notes:

1 Variables specified in this position must resolve to an OBJECT IDENTIFIER.

An application program sends an OBJECT IDENTIFIER to CMIP Services by using an explicit value. OBJECT IDENTIFIERs are specified as text strings of integers separated by periods as in 1.3.18.0.0.6. Each of the numbers of the OBJECT IDENTIFIER must resolve to a long integer. An OBJECT IDENTIFIER must contain at least two numbers in an OBJECT IDENTIFIER, but there is no maximum number of components. The first number must be either 0, 1, or 2.

How an application program specifies an OBJECT IDENTIFIER value

The following example shows how the ASN.1 syntax might define an OBJECT IDENTIFIER.

```
X ::= OBJECT IDENTIFIER

Values for X would be:
      1.2.3.4.5.6
      1.3.18.0.0.255
      2.9.3.2.6.18
```

How CMIP services sends an OBJECT IDENTIFIER to an application program

CMIP services sends an OBJECT IDENTIFIER as an explicit value.

CMIP services places labels in the string for all elements of the syntax that are present.

If CMIP services cannot decode the value, CMIP services sends the value to the application program in hexadecimal BER format, enclosed in delimiters. See "Hexadecimal BER format" on page 100 for a description.

Character string types

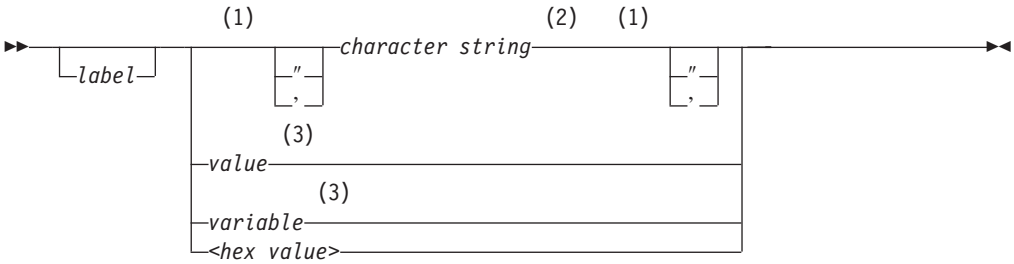
Different types of character strings can be formatted and sent to the API. Four of the string types defined in the ASN.1 standard are supported:

- NumericString
- PrintableString
- VisibleString (also known as ISO646String)
- GraphicString

The GraphicString type is the same as the ISO646String type. The application program can specify character sets other than those supported by VTAM CMIP services by using the hexadecimal BER format (see "Hexadecimal BER format" on page 100).

How an application program sends a character string to CMIP services

An application program can send a character string value to CMIP services in as normal text strings, according to the following format:



Notes:

1 Quotation marks are especially important when specifying values of character strings, because character strings are one of the few places where special characters are valid. Quotation marks are needed if any special characters such as spaces, parentheses, or commas are included in the value.

2 The characters that can be specified in this string are dictated by the ASN.1 type of the string. See the text for an explicit listing of the allowable characters.

3 Variables specified in this position must resolve to a hexadecimal or character string.

An application program can send a character string to CMIP services with or without quotation marks depending on whether the string contains special characters.

Valid characters for character strings

The characters that can be specified in the string types are defined in ISO-8824, the ASN.1 standard.

Valid characters for NumericString type

Table 4. Valid characters for NumericString

Character name	Glyph
Digits	0-9
Space	

Valid characters for PrintableString type

Table 5. Valid characters for PrintableString

Character name	Glyph
Uppercase letters	A-Z
Lowercase letters	a-z
Digits	0-9
Space	
Apostrophe	’
Left parenthesis	(
Right parenthesis	)
Plus sign	+
Comma	,
Hyphen	-
Full stop	.
Solidus	/
Colon	:
Equal sign	=
Question mark	?

Valid characters for GraphicString and ISO646String

Table 6. Valid characters for GraphicString and ISO646String

Character name	Glyph
Uppercase letters	A-Z
Lowercase letters	a-z
Digits	0-9

Table 6. Valid characters for GraphicString and ISO646String (continued)

Character name	Glyph
Space	
Exclamation mark	!
Quotation mark	'
Number sign	#
Dollar sign	$
Percent sign	%
Ampersand	&
Apostrophe	’
Left parenthesis	(
Right parenthesis	)
Asterisk	*
Plus sign	+
Comma	,
Hyphen	-
Full stop	.
Solidus	/
Colon	:
Semicolon	;
Less than sign	<
Equals sign	=
Greater than sign	>
Question mark	?
Commercial at	@
Left square bracket	[
Reverse solidus	\
Right square bracket	]
Upward arrow head	^
Underline	_
Grave accent	`
Left curly bracket	{
Vertical line	|
Right curly bracket	}
Overline	‾

Composite graphics, which are those constructed with backspaces in a GraphicString, are not allowed.

If a character that is not valid is entered on encoding, the string is rejected and an error code is returned to the application program. On decoding, characters that are not valid are accepted and translated to periods.

How CMIP services sends a character string to an application program

When CMIP services sends character strings, if the value contains the double quotation mark (") character, CMIP services encloses the value in single quotation marks. If the string does not contain the double quotation mark character, CMIP services encloses the value in double quotation marks.

CMIP services places labels in the string for all elements of the syntax that are present.

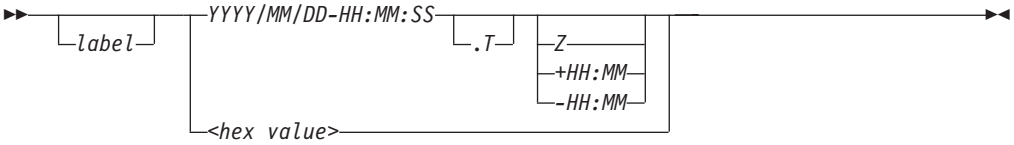
If CMIP services cannot decode the value, CMIP services sends the value to the application program in hexadecimal BER format, enclosed in delimiters. See "Hexadecimal BER format" on page 100 for a description.

Time types

Two time specifications are supported by the API: GeneralizedTime and UniversalTime.

How an application program sends a TIME value to CMIP services

An application program can send a TIME value to CMIP services for either type of time is in the following forms:



where the initial fields correspond to the year (4 digits), month, day, hours (specified using the 24-hour clock), minutes, and seconds. The additional fields are optional and can be included if the sender chooses. These represent the tenths of a second (.T), the type of the time (Z indicates GMT, + or - indicates a GMT offset and nothing indicates local time).

The entire non-hexadecimal value can be enclosed in quotation marks, as can any other string value, if the sender wishes.

How CMIP services sends a TIME value to an application program

CMIP services sends a TIME value in the same format that the application program uses to send a TIME value to CMIP services. See "Hexadecimal BER format" on page 100 for a description.

Constructed ASN.1 types

Constructed types are those that combine similar or different primitive types into ordered or unordered groups. VTAM CMIP services represents the members of constructed types by enclosing the members in parentheses. There are four constructed types. Whether a type can contain members of different types and whether order is important depends on the constructed type.

Table 7. Order and members of constructed types

Constructed type	Members	Order
SET	Members can be different types.	Order of members is not important.
SEQUENCE	Members can be different types.	Order of members is important.
SET OF	All members must be the same type.	Order of members is not important.
SEQUENCE OF	All members must be the same type.	Order of members is important.

The hexadecimal BER format can also be used for constructed types. When the hexadecimal BER format is used, either the members of the constructed type can be specified as BER or the entire contents of the SET or SEQUENCE can be specified in a single value. For example, given the following ASN.1:

```
A ::= SEQUENCE
{
    a INTEGER,
    b INTEGER,
    c INTEGER
}
```

any of the following values may be specified:

```
(a 1, b 2, c 3)
(a <01>, b <02>, c <03>)
<020101020102020103>
```

The former value specification is preferred, because CMIP services can check that the values that are specified are valid and CMIP services can construct the correct encoding of the tags, lengths, and values.

How CMIP services sends a constructed type to an application program

CMIP services sends constructed values according to the format used for the primitives that make up the constructed types. For example, if the SET value is comprised of INTEGER values, CMIP services sends the values in the same format that CMIP services sends INTEGER values. Values are enclosed in parentheses and can have commas between them.

CMIP services places labels in the string for all elements of the syntax that are present.

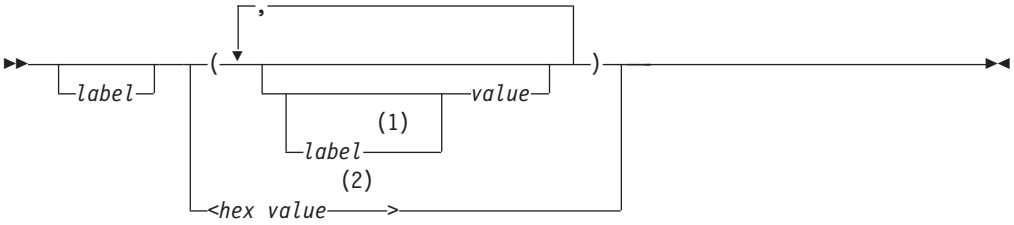
If CMIP services cannot decode the value, CMIP services sends the value to the application program in hexadecimal BER format, enclosed in delimiters. See "Hexadecimal BER format" on page 100 for a description.

For example, if an unrecognized member occurs in a SEQUENCE or a duplicate member occurs in a SET, the entire contents of the constructed type is returned as a single hexadecimal string.

SEQUENCE

A SEQUENCE is common in ASN.1. The number and order of members of the SEQUENCE are dictated by the ASN.1 definition of the SEQUENCE.

An application program can send a SEQUENCE to CMIP services in any of the following forms:



Notes:

1 Labels are required only if an optional element of the sequence is omitted and a subsequent member is included.

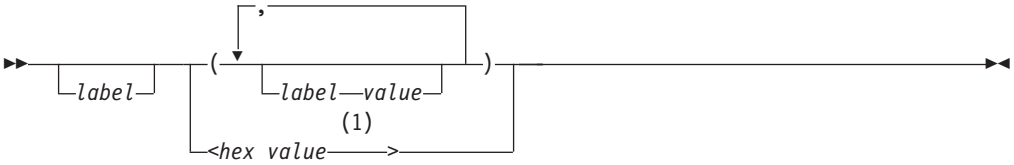
2 When specifying a value in this format, the application program is required to specify the entire contents of the SEQUENCE, including the tags and lengths of the members, but not the tag and length of the SEQUENCE itself.

Whether a particular member is required to be included depends on whether the ASN.1 definition indicates that it is optional. It does not depend on CMIP services.

SET

A SET is an unordered collections of members in ASN.1, and CMIP services implements this definition by allowing the input of members of the set in any order. Because members can be in any order, CMIP services requires that labels be specified on all SET members.

An application program can send a SET to CMIP services in the following form, which is similar to that for a SEQUENCE:



Notes:

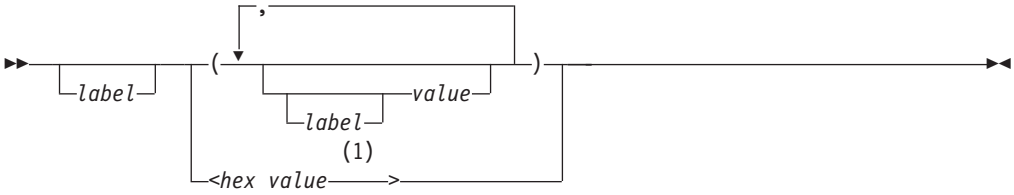
1 When specifying a value in this format, the application program is required to specify the entire contents of the SET, including the tags and lengths of the members, but not the tag and length of the SET itself.

As with a SEQUENCE, whether a particular member is required to be included is determined by whether the ASN.1 definition indicates that it is optional. It does not depend on CMIP services. Note that labels are required on members of a SET.

SET OF and SEQUENCE OF types

The SET OF and SEQUENCE OF types represent one or more instances of a SET or a SEQUENCE. For a description of the differences among constructed types, refer to Table 7 on page 113.

An application program can send a SET OF or SEQUENCE OF value to CMIP services in the following form:



Notes:

1 When specifying a value in this format, the application program is required to specify the entire contents of the SET OF or SEQUENCE OF, including the tags and lengths of the members, but not the tag and length of the SET OF or SEQUENCE OF itself.

There is no limitation (other than subtyping specified in the ASN.1) as to the number of members that can be specified in the SET OF or SEQUENCE OF. It is valid to specify a SET OF or SEQUENCE OF with no members, so long as subtype constraints are obeyed.

Decision types

Three ASN.1 types allow the application program to include different pieces of information, even after the ASN.1 definition is complete. They allow the application program to determine, at execution time, what information should fall within certain fields. VTAM CMIP services calls these types *decision types*, and they include CHOICE, ANY and ANY DEFINED BY.

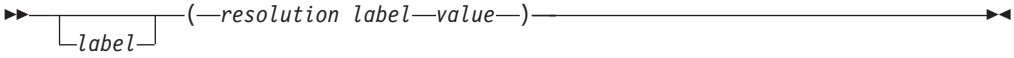
Note that the hexadecimal BER format is not supported for CHOICE and ANY DEFINED BY types. If the application program needs to specify the latter as BER, the entire SEQUENCE that contains the ANY DEFINED BY must be specified as a single BER value.

CHOICE types

A CHOICE type is one in which a decision must be made concerning the next type to include in a string. When receiving incoming strings to be decoded, the determination of which CHOICE to take is based on the tagging in the transfer syntax. In CMIP services, the choice is based on the *resolution label* presented in the string when the CHOICE is encountered. The resolution label is the identifier of each of the NamedTypes in the CHOICE construct.

How an application program sends a CHOICE to CMIP services

An application program can send a CHOICE to CMIP services in any of the following forms:



Note: The resolution label is always required.

How an application program specifies CHOICE values

The following example shows how the ASN.1 syntax might define a CHOICE.

```
A ::= CHOICE
      {
          x INTEGER,
          y OBJECT IDENTIFIER,
          z OCTET STRING
      }
```

The application program can choose to have field b be an INTEGER, an OBJECT IDENTIFIER, or an OCTET STRING. If the application program chooses for it to be an INTEGER, the following string should be specified:

```
b (x 1234)
```

where the x is the resolution label.

How CMIP services sends a CHOICE to an application program

So long as the alternative described in the BER exists within the CHOICE, CMIP services sends the CHOICE in the same format an application program uses to send a CHOICE to CMIP services. (An alternative is one of the options specified in the CHOICE syntax.)

CMIP services places labels in the string for all elements of the syntax that are present.

If CMIP services cannot decode the value, CMIP services sends the value to the application program in hexadecimal BER format, enclosed in delimiters. See "Hexadecimal BER format" on page 100 for a description. For example, if CMIP services does not recognize the alternative, CMIP services sends a CHOICE as hexadecimal BER.

ANY DEFINED BY types

How an application program sends an ANY DEFINED BY value to CMIP services

An application program can send an ANY DEFINED BY value to CMIP services according to the method used to send the type to which the ANY DEFINED BY resolves. The label of the input corresponds to the label of the ANY DEFINED BY construct in the ASN.1, and the value corresponds to the value of the type to which the ANY DEFINED BY resolves. The resolution field determines which type to translate.

How an application program specifies ANY DEFINED BY values

The following example shows how the ASN.1 syntax might define an ANY DEFINED BY value.

```
    A ::= INTEGER
    B ::= BIT STRING
    C ::= BOOLEAN

    X ::= SEQUENCE
          {
              a INTEGER,
              b ANY DEFINED BY a --% ANY_TABLE_REF (Y)
          }
--% Y ANY_TABLE ::=
--%       {
--%           1  A,
--%           2  B,
--%           3  C
--%       }
```

Given this ASN.1, if one wanted to have member *b* of type *X* be a bit string, field *a* must have a value of 2 (as defined by the ANY DEFINED BY resolution table Y). Therefore, an application program formats and sends to CMIP services the following:

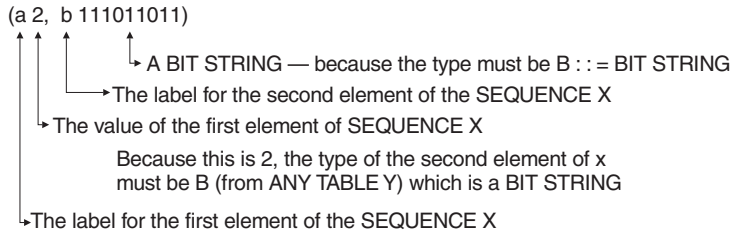


Figure 4. Defining a bit string field

ANY types

The ANY type in ASN.1 carries no tagging information and can resolve to any other ASN.1 type. Because an unknown set of different types can be used to resolve an ANY, the API must be told about the tag to be used.

How an application program sends an ANY value to CMIP services

An application program can send an ANY value to CMIP services in the following form:

Notes:

1 The hexadecimal value specified in this position must include the tag and length fields of the BER. Note that this is different from the hexadecimal values specified for other types.

The only valid format for an application program to use for an ANY value is hexadecimal BER.

How CMIP services sends an ANY value to an application program

CMIP services sends an ANY value in hexadecimal BER format. It is important to note that the hexadecimal value for an ANY value **includes** the tag and length portions of the BER, in contrast to the hexadecimal BER formats of the other types.

CMIP services places labels in the string for all elements of the syntax that are present.

Additional examples of how application programs send data

The following examples demonstrate how an application program can send primitive types and the more complex ASN.1 data types.

The first examples are based on the following ASN.1 module:

```
Abc DEFINITIONS IMPLICIT TAGS ::= BEGIN
    A ::= BOOLEAN
    B ::= INTEGER
```

```
    C ::= ENUMERATED {a(0), b(2), c(5), d(10)}
    D ::= REAL
    E ::= BIT STRING
    F ::= OCTET STRING
    G ::= NULL
    H ::= OBJECT IDENTIFIER

    a A ::= TRUE
    b B ::= 12
    c C ::= 10
    e E ::= B'10010'
    f F ::= H'1234567890'
    g G ::= NULL
    h H ::= { iso icd(3) 18 0 0 6 }
END
```

The following are all valid input strings:

Module	Type	String
Abc	A	TRUE
Abc	A	false
Abc	A	a
Abc	B	-12345
Abc	B	0
Abc	B	500000
Abc	B	b
Abc	C	b
Abc	C	c
Abc	D	3.125
Abc	D	-12E25
Abc	E	11011011010
Abc	E	""
Abc	E	e
Abc	F	1234567890123456
Abc	F	f
Abc	F	""
Abc	G	NULL
Abc	G	g
Abc	H	1.3.18.0.3
Abc	H	0.0
Abc	H	1.2.5.355465.2.1
Abc	H	h

The second set of examples show how to specify some constructed data types. This set of examples is based on the following ASN.1 module:

```
Xyz DEFINITIONS IMPLICIT TAGS ::= BEGIN

    X ::= SEQUENCE
          {
              a INTEGER,
              b BOOLEAN OPTIONAL,
```

```
            c INTEGER,
            d BIT STRING
        }
    Y ::= SET OF INTEGER
    Z ::= SEQUENCE
        {
            a X,
            b Y
        }

END
```

The following are all valid strings that the application program can send to CMIP services.

Module	Type	String
Xyz	X	(a 12, b TRUE, c 56000, d 1101101)
Xyz	X	(a 12, c 56000, d “1101101”)
Xyz	X	(12, FALSE, 0, "")
Xyz	Y	(1, 2, 3, 4, 5, 6, 7, 8)
Xyz	Y	(1,2,3,4,5,6)
Xyz	Y	()
Xyz	Z	(a (a 12, b TRUE, c 56000, d 1101101), b (1,2,3,4))
Xyz	Z	(a (a 12, c 56000, d 1101101), b ())
Xyz	Z	((a 12, c 56000, d 1101101), ())

Chapter 8. Examples of standard CMIP strings

This section contains examples of the CMIP strings that are sent between application programs and CMIP services.

The requests and responses are sent from the application program to CMIP services using the MIBSendCmipRequest and MIBSendCmipResponse functions. For a description of these functions, refer to pages "MIBSendCmipRequest—CMIP request function" on page 70 and "MIBSendCmipResponse—CMIP response function" on page 73. The indications and confirmations are received by the application program using the read queue exit routine or the dataspace dequeue routine.

The following example shows the call that an application program makes to the MIBSendCmipRequest to send a CMIP request. The values for the variables OperationValue and Argument will be determined by the type of request being sent. Examples on the following pages will show specific examples for the values of these variables.

```
int LinkId;
int rc;
void *LocalId;
unsigned int InvokeId;
unsigned int OperationValue;
char Argument[4096];

rc = MIBSendCmipRequest(LinkId,
                        OperationValue, /* 3 for GET, 7 for Action,
                                           8 for CREATE, etc.           */
                        Argument,
                        LocalId,
                        NULL,     /* don't override source object
                                     specified by LocalId               */
                        DS_NOT_PROVIDED, /* don't override dest         */
                        NULL,
                        &InvokeId);
```

The following example shows the call that an application program makes to the MIBSendCmipResponse to send a CMIP response. The values for the variables OperationValue and Argument will be determined by the type of response being sent. Examples on the following pages will show specific examples for the values of these variables.

```
char *AssocHandleFromRequest;
int LinkId;
int rc;
void *LocalId;
unsigned int InvokeId, InvokeIdFromRequest;
unsigned int OperationValue;
char Argument[4096];

rc = MIBSendCmipResponse(LinkId,
                         InvokeIdFromRequest,
                         1,       /* last-in-chain indicator       */
                         1,       /* successful                    */
                         OperationValue, /* 3 for GET, 7 for Action,
                                            8 for CREATE, etc.       */
                         Argument,
                         LocalId,
```

```
NULL,    /* don't override source object
            specified by LocalId          */
AssocHandleFromRequest,
&InvokeId);
```

Requests and indications

The following descriptions are for CMIP requests and indications. A request is the message sent by a manager application program to an agent application program via the MIBSendCmipRequest function.

An indication is the message received by the agent application program corresponding to the request.

For each request, the following information is included:
- ASN.1 syntax
- Example request string
- Corresponding indication

GET request—syntax

```
GetArgument ::= SEQUENCE
    {
        baseManagedObjectClass       ObjectClass,
        baseManagedObjectInstance    ObjectInstance,
        accessControl            [5] AccessControl OPTIONAL,
        synchronization          [6] IMPLICIT CMISSync DEFAULT bestEffort,
        scope                    [7] Scope      DEFAULT base-and : baseObject,
        filter                       CMISFilter DEFAULT and : {},
        attributeIdList         [12] IMPLICIT SET OF AttributeId OPTIONAL
    }
```

GET request—example request string

The operation-value for GET is 3, so the value of the OperationValue variable will be 3 as well.

Here is an example value of the Argument variable:

```
(baseManagedObjectClass 2.9.3.2.3.13,baseManagedObjectInstan
ce (distinguishedName "1.3.18.0.2.4.6=NETA;2.9.3.2.7.4=(name
 SSCP1A)"))
```

This will retrieve the values of all attributes of the system object on host NETA.SSCP1A.

GET request—corresponding indication

Here is an example GET indication corresponding to the previous GET request example, as received by the application. This shows the APIhdr at the beginning of the message.

```
00000100 0003000A 00000001 00000001    *.............*
2FAA4356 00000000 00000000 01120020    *.............*
A2998360 A3A89785 40F16B40 A2998340    *src-type 1, src *
81F16B40 94A28740 C3D4C9D7 60F14BD5    *a1, msg CMIP-1.N*
96A38986 898381A3 89969540 4D8995A5    *otification (inv*
969285C9 C440F1F9 F6F6F1F8 6B409697    *okeID 196618, op*
859981A3 89969560 A58193A4 8540F06B    *eration-value 0,*
40819987 A4948595 A3404D94 81958187    * argument (manag*
8584D682 918583A3 C39381A2 A240F14B    *edObjectClass 1.*
F34BF1F8 4BF04BF0 4BF2F2F6 F76B4094    *3.18.0.0.2267, m*
81958187 8584D682 918583A3 C995A2A3    *anagedObjectInst*
```

```
81958385 404D8489 A2A38995 87A489A2    *ance (distinguis*
888584D5 81948540 7DF14BF3 4BF1F84B    *hedName '1.3.18.*
F04BF24B F44BF67E D5C5E3C1 5EF14BF3    *0.2.4.6=NETA;1.3*
4BF1F84B F04BF04B F2F0F3F2 7EE2E2C3    *.18.0.0.2032=SSC*
D7F1C15E F14BF34B F1F84BF0 4BF04BF2    *P1A;1.3.18.0.0.2*
F2F7F27E E2E6C9E3 C3C8C5C4 4BE2E6D5    *272=SWITCHED.SWN*
C4F3C1C2 F77D5D6B 4085A585 95A3E3A8    *D3AB7'), eventTy*
978540F2 4BF94BF3 4BF24BF1 F04BF65D    *pe 2.9.3.2.10.6)*
5D00                                   *).             *
```

ACTION request—syntax

```
ActionArgument ::= SEQUENCE {
       baseManagedObjectClass       ObjectClass,
       baseManagedObjectInstance    ObjectInstance,
       accessControl            [5] AccessControl OPTIONAL,
       synchronization          [6] IMPLICIT CMISSync DEFAULT bestEffort,
       scope                    [7] Scope OPTIONAL,
       filter                       CMISFilter DEFAULT and : {},
       actionInfo              [12] IMPLICIT ActionInfo
       }
```

ACTION request—example request string

The operation-value for ACTION is 7, so the value of the OperationValue variable will be 7 as well.

Here is an example value of the Argument variable:

```
(baseManagedObjectClass 1.3.18.0.0.2151,baseManagedObjectIns
tance (distinguishedName "1.3.18.0.2.4.6=NETA;1.3.18.0.0.203
2=SSCP1A;1.3.18.0.0.2216=(string SnaNetwork)"),actionInfo (a
ctionType 1.3.18.0.0.2222, actionInfoArg (start oneTimeOnly)
))
```

ACTION request—corresponding indication

Here is an example ACTION indication corresponding to the previous ACTION request example, as received by the application. This shows the APIhdr at the beginning of the message.

```
                  00000100 00030012    *        ........*
00000002 00000001 2FAA536E 00000000    *...........>....*
00000000 0120FA08 A2998360 A3A89785    *........src-type*
40F16B40 A2998340 81F16B40 94A28740    * 1, src a1, msg *
C3D4C9D7 60F14BD9 D6C9E581 9784A440    *CMIP-1.ROIVapdu *
4D8995A5 969285C9 C440F1F9 F6F6F2F6    *(invokeID 196626*
6B409697 859981A3 89969560 A58193A4    *, operation-valu*
8540F76B 40819987 A4948595 A3404D82    *e 7, argument (b*
81A285D4 81958187 8584D682 918583A3    *aseManagedObject*
C39381A2 A240F14B F34BF1F8 4BF04BF0    *Class 1.3.18.0.0*
4BF2F1F5 F16B4082 81A285D4 81958187    *.2151, baseManag*
8584D682 918583A3 C995A2A3 81958385    *edObjectInstance*
404D8489 A2A38995 87A489A2 888584D5    * (distinguishedN*
81948540 4DD98593 81A389A5 85C489A2    *ame (RelativeDis*
A3899587 A489A288 8584D581 9485404D    *tinguishedName (*
C1A3A399 8982A4A3 85E58193 A485C1A2    *AttributeValueAs*
A28599A3 89969540 4D81A3A3 998982A4    *sertion (attribu*
A385E3A8 978540F1 4BF34BF1 F84BF04B    *teType 1.3.18.0.*
F24BF44B F66B4081 A3A39989 82A4A385    *2.4.6, attribute*
E58193A4 85407FD5 C5E3C17F 5D5D6B40    *Value "NETA")), *
D9859381 A389A585 C489A2A3 899587A4    *RelativeDistingu*
89A28885 84D58194 85404DC1 A3A39989    *ishedName (Attri*
82A4A385 E58193A4 85C1A2A2 8599A389    *buteValueAsserti*
9695404D 81A3A399 8982A4A3 85E3A897    *on (attributeTyp*
8540F14B F34BF1F8 4BF04BF0 4BF2F0F3    *e 1.3.18.0.0.203*
F26B4081 A3A39989 82A4A385 E58193A4    *2, attributeValu*
```

```
85407FE2 E2C3D7F1 C17F5D5D 6B40D985    *e "SSCP1A")), Re*
9381A389 A585C489 A2A38995 87A489A2    *lativeDistinguis*
888584D5 81948540 4DC1A3A3 998982A4    *hedName (Attribu*
A385E581 93A485C1 A2A28599 A3899695    *teValueAssertion*
404D81A3 A3998982 A4A385E3 A8978540    * (attributeType *
F14BF34B F1F84BF0 4BF04BF2 F2F1F66B    *1.3.18.0.0.2216,*
4081A3A3 998982A4 A385E581 93A48540    * attributeValue *
4DA2A399 89958740 7FE29581 D585A3A6    *(string "SnaNetw*
9699927F 5D5D5D5D 5D6B4081 83A38996    *ork")))))), actio*
95C99586 96404D81 83A38996 95E3A897    *nInfo (actionTyp*
8540F14B F34BF1F8 4BF04BF0 4BF2F2F2    *e 1.3.18.0.0.222*
F26B4081 83A38996 95C99586 96C19987    *2, actionInfoArg*
404DA2A3 8199A340 969585E3 899485D6    * (start oneTimeO*
9593A85D 5D5D5D00                      *nly))))).       *
```

Responses and confirmations

The following descriptions are for CMIP responses and confirmations. A response is the message sent by an agent application program to a manager application program via the MIBSendCmipResponse function.

A confirmation is the message received by the manager application program which corresponds to the response.

For each response or confirmation, the following information is included:
- ASN.1 syntax
- Example response string
- Corresponding confirmation

GET response—syntax

```
GetResult::=
      SEQUENCE { managedObjectClass ObjectClass OPTIONAL,
                 managedObjectInstance ObjectInstance OPTIONAL,
                 currentTime [5] IMPLICIT GeneralizedTime OPTIONAL,
                 attributeList [6] IMPLICIT SET OF Attribute OPTIONAL
               }
```

GET response—example response string

The operation-value for GET is 3, so the value of the OperationValue variable will be 3 as well.

Here is an example value of the Argument variable:

```
(managedObjectClass &OC,(distinguishedName &DN),attri
buteList ((attributeId 2.9.3.2.7.5, (distinguishedNam
e (((attributeType     1.3.18.0.2.4.6,attributeValue
NETA), (attributeType 2.9.3.2.7.4,attributeValue
(name "SSCP1A")))))), (attributeId 2.9.3.2.7.35, enab
led)))
```

GET response—corresponding confirmation

Here is an example GET confirmation corresponding to the previous GET response example, as received by the application. This shows the APIhdr at the beginning of the message.

```
                  00000000 00030008    *        ........*
00000003 00000001 2FB39434 00000000    *..........m.....*
00000000 00000001 A2998360 A3A89785    *........src-type*
40F16B40 A2998340 81F16B40 94A28740    * 1, src a1, msg *
C3D4C9D7 60F14BD9 D6D9E281 9784A440    *CMIP-1.RORSapdu *
4D8995A5 969285C9 C440F1F9 F6F6F1F6    *(invokeID 196616*
```

```
6B409985 A2A493A3 D697A389 9695404D    *, resultOption (*
96978599 81A38996 9560A581 93A48540    *operation-value *
F36B4099 85A2A493 A3404D94 81958187    *3, result (manag*
8584D682 918583A3 C39381A2 A240F24B    *edObjectClass 2.*
F94BF34B F24BF34B F1F36B40 94819581    *9.3.2.3.13, mana*
878584D6 82918583 A3C995A2 A3819583    *gedObjectInstanc*
85404D84 89A2A389 9587A489 A2888584    *e (distinguished*
D5819485 404DD985 9381A389 A585C489    *Name (RelativeDi*
A2A38995 87A489A2 888584D5 81948540    *stinguishedName *
4DC1A3A3 998982A4 A385E581 93A485C1    *(AttributeValueA*
A2A28599 A3899695 404D81A3 A3998982    *ssertion (attrib*
A4A385E3 A8978540 F14BF34B F1F84BF0    *uteType 1.3.18.0*
4BF24BF4 4BF66B40 81A3A399 8982A4A3    *.2.4.6, attribut*
85E58193 A485407F D5C5E3C1 7F5D5D6B    *eValue "NETA")),*
40D98593 81A389A5 85C489A2 A3899587    * RelativeDisting*
A489A288 8584D581 9485404D C1A3A399    *uishedName (Attr*
8982A4A3 85E58193 A485C1A2 A28599A3    *ibuteValueAssert*
89969540 4D81A3A3 998982A4 A385E3A8    *ion (attributeTy*
978540F2 4BF94BF3 4BF24BF7 4BF46B40    *pe 2.9.3.2.7.4, *
81A3A399 8982A4A3 85E58193 A485404D    *attributeValue (*
95819485 407FE2E2 C3D7F1C1 7F5D5D5D    *name "SSCP1A")))*
5D5D6B40 81A3A399 8982A4A3 85D389A2    *)), attributeLis*
A3404DC1 A3A39989 82A4A385 404D81A3    *t (Attribute (at*
A3998982 A4A385C9 8440F24B F94BF34B    *tributeId 2.9.3.*
F24BF74B F56B4081 A3A39989 82A4A385    *2.7.5, attribute*
E58193A4 85404D84 89A2A389 9587A489    *Value (distingui*
A2888584 D5819485 404DD985 9381A389    *shedName (Relati*
A585C489 A2A38995 87A489A2 888584D5    *veDistinguishedN*
81948540 4DC1A3A3 998982A4 A385E581    *ame (AttributeVa*
93A485C1 A2A28599 A3899695 404D81A3    *lueAssertion (at*
A3998982 A4A385E3 A8978540 F14BF34B    *tributeType 1.3.*
F1F84BF0 4BF24BF4 4BF66B40 81A3A399    *18.0.2.4.6, attr*
8982A4A3 85E58193 A485407F D5C5E3C1    *ibuteValue "NETA*
7F5D6B40 C1A3A399 8982A4A3 85E58193    *"), AttributeVal*
A485C1A2 A28599A3 89969540 4D81A3A3    *ueAssertion (att*
998982A4 A385E3A8 978540F2 4BF94BF3    *ributeType 2.9.3*
4BF24BF7 4BF46B40 81A3A399 8982A4A3    *.2.7.4, attribut*
85E58193 A485404D 95819485 407FE2E2    *eValue (name "SS*
C3D7F1C1 7F5D5D5D 5D5D6B40 C1A3A3      *CP1A")))))), Att*
998982A4 A385404D 81A3A399 8982A4A3    *ribute (attribut*
85C98440 F24BF94B F34BF24B F74BF3F5    *eId 2.9.3.2.7.35*
6B4081A3 A3998982 A4A385E5 8193A485    *, attributeValue*
40859581 82938584 5D5D5D5D 5D00        * enabled))))).  *
```

CREATE response—syntax

```
CreateResult::=
        SEQUENCE { managedObjectClass ObjectClass OPTIONAL,
                   managedObjectInstance ObjectInstance OPTIONAL,
                   currentTime [5] IMPLICIT GeneralizedTime OPTIONAL,
                   attributeList [6] IMPLICIT SET OF Attribute OPTIONAL
                   }
```

CREATE response—example response string

The operation-value for CREATE is 8, so the value of the OperationValue variable will be 8 as well.

Here is an example value of the Argument variable:

```
(managedObjectClass 1.3.18.0.0.2054, (distinguis
hedName '1.3.18.0.2.4.6=NETA;2.9.3.2.7.4=(name "
SSCP1A");2.9.3.2.7.1=(string "EFD00001")'), attr
ibuteList ((2.9.3.2.7.65, 1.3.18.0.0.2),(2.9.3.2
.7.1, (string "EFD00001")),(2.9.3.2.7.66, (2.9.3
.2.4.17,2.9.3.2.4.22,1.3.18.0.0.2063)),(2.9.3.2.
7.50, (2.9.3.2.3.4)),(2.9.3.2.7.31, unlocked),(2
```

```
.9.3.2.7.35, enabled),(1.3.18.0.0.2775, ()),(2.9
.3.2.7.33, ()),(2.9.3.2.7.56,(item (equality (at
tributeId 2.9.3.2.7.14, attributeValue 2.9.3.2.1
0.7)))),(2.9.3.2.7.55,(single (name (RDNSequence
 (RelativeDistinguishedName (AttributeValueAsser
tion (attributeType 1.3.18.0.2.4.6, attributeVal
ue "NETA")), RelativeDistinguishedName (Attribut
eValueAssertion (attributeType 2.9.3.2.7.4, attr
ibuteValue (name "SSCP1A"))), RelativeDistinguis
hedName (AttributeValueAssertion (attributeType
1.3.18.0.2.4.12, attributeValue "aplposec"))))))
),(2.9.3.2.7.63,2.9.3.2.6.1)))
```

CREATE response—corresponding confirmation

Here is an example CREATE confirmation corresponding to the previous CREATE response example, as received by the application. This shows the APIhdr at the beginning of the message.

```
                  00000000 0003000A    *        ........*
00000003 00000001 2FB398C6 00000000    *..........qF....*
00000000 00000004 A2998360 A3A89785    *........src-type*
40F16B40 A2998340 81F16B40 94A28740    * 1, src a1, msg *
C3D4C9D7 60F14BD9 D6D9E281 9784A440    *CMIP-1.RORSapdu *
4D8995A5 969285C9 C440F1F9 F6F6F1F8    *(invokeID 196618*
6B409985 A2A493A3 D697A389 9695404D    *, resultOption (*
96978599 81A38996 9560A581 93A48540    *operation-value *
F86B4099 85A2A493 A3404D94 81958187    *8, result (manag*
8584D682 918583A3 C39381A2 A240F14B    *edObjectClass 1.*
F34BF1F8 4BF04BF0 4BF2F0F5 F46B4094    *3.18.0.0.2054, m*
81958187 8584D682 918583A3 C995A2A3    *anagedObjectInst*
81958385 404D8489 A2A38995 87A489A2    *ance (distinguis*
888584D5 81948540 4DD98593 81A389A5    *hedName (Relativ*
85C489A2 A3899587 A489A288 8584D581    *eDistinguishedNa*
9485404D C1A3A399 8982A4A3 85E58193    *me (AttributeVal*
A485C1A2 A28599A3 89969540 4D81A3A3    *ueAssertion (att*
998982A4 A385E3A8 978540F1 4BF34BF1    *ributeType 1.3.1*
F84BF04B F24BF44B F66B4081 A3A39989    *8.0.2.4.6, attri*
82A4A385 E58193A4 85407FD5 C5E3C17F    *buteValue "NETA"*
5D5D6B40 D9859381 A389A585 C489A2A3    *)), RelativeDist*
899587A4 89A28885 84D58194 85404DC1    *inguishedName (A*
A3A39989 82A4A385 E58193A4 85C1A2A2    *ttributeValueAss*
8599A389 9695404D 81A3A399 8982A4A3    *ertion (attribut*
85E3A897 8540F24B F94BF34B F24BF74B    *eType 2.9.3.2.7.*
F46B4081 A3A39989 82A4A385 E58193A4    *4, attributeValu*
85404D95 81948540 7FE2E2C3 D7F1C17F    *e (name "SSCP1A"*
5D5D5D6B 40D98593 81A389A5 85C489A2    *))), RelativeDis*
A3899587 A489A288 8584D581 9485404D    *tinguishedName (*
C1A3A399 8982A4A3 85E58193 A485C1A2    *AttributeValueAs*
A28599A3 89969540 4D81A3A3 998982A4    *sertion (attribu*
A385E3A8 978540F2 4BF94BF3 4BF24BF7    *teType 2.9.3.2.7*
4BF16B40 81A3A399 8982A4A3 85E58193    *.1, attributeVal*
A485404D A2A39989 9587407F C5C6C4F0    *ue (string "EFD0*
F0F0F0F1 7F5D5D5D 5D5D6B40 81A3A399    *0001")))), attr*
8982A4A3 85D389A2 A3404DC1 A3A39989    *ibuteList (Attri*
82A4A385 404D81A3 A3998982 A4A385C9    *bute (attributeI*
8440F24B F94BF34B F24BF74B F6F56B40    *d 2.9.3.2.7.65, *
81A3A399 8982A4A3 85E58193 A48540F1    *attributeValue 1*
4BF34BF1 F84BF04B F04BF25D 6B40C1A3    *.3.18.0.0.2), At*
A3998982 A4A38540 4D81A3A3 998982A4    *tribute (attribu*
A385C984 40F24BF9 4BF34BF2 4BF74BF1    *teId 2.9.3.2.7.1*
6B4081A3 A3998982 A4A385E5 8193A485    *, attributeValue*
404DA2A3 99899587 407FC5C6 C4F0F0F0    * (string "EFD000*
F0F17F5D 5D6B40C1 A3A39989 82A4A385    *01")), Attribute*
404D81A3 A3998982 A4A385C9 8440F24B    * (attributeId 2.*
F94BF34B F24BF74B F6F66B40 81A3A399    *9.3.2.7.66, attr*
8982A4A3 85E58193 A485404D D6C2D1C5    *ibuteValue (OBJE*
```

```
C3E360C9 C4C5D5E3 C9C6C9C5 D940F24B    *CT-IDENTIFIER 2.*
F94BF34B F24BF44B F1F76B40 D6C2D1C5    *9.3.2.4.17, OBJE*
C3E360C9 C4C5D5E3 C9C6C9C5 D940F24B    *CT-IDENTIFIER 2.*
F94BF34B F24BF44B F2F26B40 D6C2D1C5    *9.3.2.4.22, OBJE*
C3E360C9 C4C5D5E3 C9C6C9C5 D940F14B    *CT-IDENTIFIER 1.*
F34BF1F8 4BF04BF0 4BF2F0F6 F35D5D6B    *3.18.0.0.2063)),*
40C1A3A3 998982A4 A385404D 81A3A399    * Attribute (attr*
8982A4A3 85C98440 F24BF94B F34BF24B    *ibuteId 2.9.3.2.*
F74BF5F0 6B4081A3 A3998982 A4A385E5    *7.50, attributeV*
8193A485 404DD682 918583A3 C39381A2    *alue (ObjectClas*
A240F24B F94BF34B F24BF34B F45D5D6B    *s 2.9.3.2.3.4)),*
40C1A3A3 998982A4 A385404D 81A3A399    * Attribute (attr*
8982A4A3 85C98440 F24BF94B F34BF24B    *ibuteId 2.9.3.2.*
F74BF3F1 6B4081A3 A3998982 A4A385E5    *7.31, attributeV*
8193A485 40A49593 96839285 845D6B40    *alue unlocked), *
C1A3A399 8982A4A3 85404D81 A3A39989    *Attribute (attri*
82A4A385 C98440F2 4BF94BF3 4BF24BF7    *buteId 2.9.3.2.7*
4BF3F56B 4081A3A3 998982A4 A385E581    *.35, attributeVa*
93A48540 85958182 9385845D 6B40C1A3    *lue enabled), At*
A3998982 A4A38540 4D81A3A3 998982A4    *tribute (attribu*
A385C984 40F14BF3 4BF1F84B F04BF04B    *teId 1.3.18.0.0.*
F2F7F7F5 6B4081A3 A3998982 A4A385E5    *2775, attributeV*
8193A485 404D5D5D 6B40C1A3 A3998982    *alue ()), Attrib*
A4A38540 4D81A3A3 998982A4 A385C984    *ute (attributeId*
40F24BF9 4BF34BF2 4BF74BF3 F36B4081    * 2.9.3.2.7.33, a*
A3A39989 82A4A385 E58193A4 85404D5D    *ttributeValue ()*
5D6B40C1 A3A39989 82A4A385 404D81A3    *), Attribute (at*
A3998982 A4A385C9 8440F24B F94BF34B    *tributeId 2.9.3.*
F24BF74B F5F66B40 81A3A399 8982A4A3    *2.7.56, attribut*
85E58193 A485404D 89A38594 404D8598    *eValue (item (eq*
A4819389 A3A8404D 81A3A399 8982A4A3    *uality (attribut*
85C98440 F24BF94B F34BF24B F74BF1F4    *eId 2.9.3.2.7.14*
6B4081A3 A3998982 A4A385E5 8193A485    *, attributeValue*
40F24BF9 4BF34BF2 4BF1F04B F75D5D5D    * 2.9.3.2.10.7)))*
5D6B40C1 A3A39989 82A4A385 404D81A3    *), Attribute (at*
A3998982 A4A385C9 8440F24B F94BF34B    *tributeId 2.9.3.*
F24BF74B F5F56B40 81A3A399 8982A4A3    *2.7.55, attribut*
85E58193 A485404D A2899587 9385404D    *eValue (single (*
95819485 404DD9C4 D5E28598 A4859583    *name (RDNSequenc*
85404DD9 859381A3 89A585C4 89A2A389    *e (RelativeDisti*
9587A489 A2888584 D5819485 404DC1A3    *nguishedName (At*
A3998982 A4A385E5 8193A485 C1A2A285    *tributeValueAsse*
99A38996 95404D81 A3A39989 82A4A385    *rtion (attribute*
E3A89785 40F14BF3 4BF1F84B F04BF24B    *Type 1.3.18.0.2.*
F44BF66B 4081A3A3 998982A4 A385E581    *4.6, attributeVa*
93A48540 7FD5C5E3 C17F5D5D 6B40D985    *lue "NETA")), Re*
9381A389 A585C489 A2A38995 87A489A2    *lativeDistinguis*
888584D5 81948540 4DC1A3A3 998982A4    *hedName (Attribu*
A385E581 93A485C1 A2A28599 A3899695    *teValueAssertion*
404D81A3 A3998982 A4A385E3 A8978540    * (attributeType *
F24BF94B F34BF24B F74BF46B 4081A3A3    *2.9.3.2.7.4, att*
998982A4 A385E581 93A48540 4D958194    *ributeValue (nam*
85407FE2 E2C3D7F1 C17F5D5D 5D6B40D9    *e "SSCP1A"))), R*
859381A3 89A585C4 89A2A389 9587A489    *elativeDistingui*
A2888584 D5819485 404DC1A3 A3998982    *shedName (Attrib*
A4A385E5 8193A485 C1A2A285 99A38996    *uteValueAssertio*
95404D81 A3A39989 82A4A385 E3A89785    *n (attributeType*
40F14BF3 4BF1F84B F04BF24B F44BF1F2    * 1.3.18.0.2.4.12*
6B4081A3 A3998982 A4A385E5 8193A485    *, attributeValue*
407F8197 939796A2 85837F5D 5D5D5D5D    * "aplposec")))))*
5D5D6B40 C1A3A399 8982A4A3 85404D81    *)), Attribute (a*
A3A39989 82A4A385 C98440F2 4BF94BF3    *ttributeId 2.9.3*
4BF24BF7 4BF6F36B 4081A3A3 998982A4    *.2.7.63, attribu*
A385E581 93A48540 F24BF94B F34BF24B    *teValue 2.9.3.2.*
F64BF15D 5D5D5D5D 00                   *6.1)))))..      *
```

Chapter 9. Create and delete requests

This chapter describes how an application program uses CMIP services to remotely create and delete objects on agent systems.

Objects are not directly created or deleted by CMIP services in response to CMIP m-Create and m-Delete requests. When a manager application program sends a create or delete request to an agent system, these requests are processed by CMIP agent application programs.

Create requests

CMIP services requires that the create request provide the distinguished name of the object being created.

For an object to be created by CMIP services, the name binding to be used for the object must explicitly specify that the create operation is supported. If the name binding does not explicitly specify that the create operation is supported, the create request is rejected.

Because objects are not directly created by CMIP services, an application program must exist that is capable of processing the create request.

CMIP services looks for an application program to handle the create request; this application program is called a *create handler.*

- If CMIP services *finds a create handler*, CMIP services sends the create request to the create handler.
- If CMIP services *cannot find a create handler*, CMIP services rejects the create request with a noSuchObjectClass error.

When the create handler receives the create request, it does one of the following:

- Creates the new object requested on the create request
- Rejects the create request for the new object
- Creates an object different from the object requested on the create request

Creating the new object requested on the create request

To create a new object that is to be registered on the same connection as the create handler, the create handler registers the new object with the MIBSendRegister function using the same distinguished name and object class that were specified on the create request.

After the create handler registers the new object, the create handler acknowledges the create request. The create handler uses the MIBSendCmipResponse function to return the response to the sender of the create request.

Rejecting the create request

If the create handler decides to reject the create request, the create handler uses the MIBSendDeleteRegistration function with no local identifier and the object name provided with the create request to remove the pending registration for object that was requested to be created.

Then the create handler uses the MIBSendCmipResponse function to return an error response to the sender of of the create request. The error describes to the manager application program why the create request was rejected.

Creating an object different from object on the create request

If the create handler decides to create an object different from the one that was requested to be created, the create handler uses the MIBSendDeleteRegistration function with no local identifier and the object name provided with the create request to remove the pending registration for object that was requested to be created. Then the create handler registers the other object with the MIBSendRegister function.

After the create handler registers the new object, the create handler acknowledges the create request. The create handler uses the MIBSendCmipResponse function to return the response to the sender of the create request.

Delete requests

Because objects are not directly deleted by CMIP services, all application programs must be able to handle delete requests.

For registered objects, the application program sends the delete request to the application program that registered the object. For objects that are not registered, the application program sends the delete request to the subtree manager of the object. The create handler is not involved in the processing of the delete request.

When an application program receives the delete request, it either deletes the object or rejects the delete request. These two situations are described here for non-scoped delete requests.

Deleting the object requested on the delete request

In this situation, a manager application program requests that an object be deleted and the agent application program that owns the object allows it to be deleted. In general, these are the steps that are followed:

1. The manager application program issues the CMIP delete request for an object.
2. CMIP services sends an ROIV message to the agent application program that owns the object.
3. The agent application program sends the MIB.DeleteResponse with a result code of 0 to CMIP services.
4. CMIP services sends MIB.Delete with an action code of 0 to the agent application program.
5. The agent application program uses the MIBSendCmipResponse to return the CMIP delete response to CMIP services.
6. CMIP services sends an RORS to the manager application program containing the application program's delete response.
7. CMIP services sends the API_TERMINATE_INSTANCE to the deleted object.

Rejecting the delete request

In this situation, a manager application program requests that an object be deleted and the agent application program that owns the object rejects the delete request. In general, these are the steps that are followed:

1. The manager application program issues the CMIP delete request for an object.

2. CMIP services sends an ROIV message to the agent application program that owns the object.
3. The agent application program sends the MIB.DeleteResponse with a result code of 1 to CMIP services.
4. CMIP services sends an ROER to the manager application program.

Subtree managers might receive deletes that were not scoped specifically to the subtree manager object but that might apply to an object under the subtree manager. The subtree manager must perform delete processing with its objects.

Chapter 10. VTAM-specific requests and responses

The following VTAM-specific requests and responses are accepted and processed by VTAM CMIP services. The requests are sent by the MIBSendRequest function. The responses are sent from CMIP Services to the application program. These requests and responses allow the application program to perform certain actions that are specific to VTAM CMIP services, such as:

- Subscribing to association information
- Registering an application entity title
- Starting associations
- Ending associations
- Getting association information
- Creating a dedicated association

Here are the requests and responses:

- ACF.Subscribe
- ACF.UnSubscribe
- ACF.RegisterAE
- ACF.Associate
- ACF.Release
- ACF.Abort
- ACF.GetAssociationInfo
- ACF.AssociateRsp
- ACF.SubscribeRsp
- ACF.SubscribeMess
- MIB.GeneralRequest
- MIB.GeneralResponse
- MIB.GeneralError
- MIB.ServiceError
- MIB.ServiceAccept
- MIB.RegisterAccept

In the following sections, please note that the example strings are divided across multiple lines for legibility only. The actual strings being sent must be continuous.

Subscribing to association information

The ACF.Subscribe and ACF.UnSubscribe strings cause CMIP services to notify an application program when the state of an association changes. These strings are used only when an application program depends on maintaining a connection with another application program. Because associations are automatically started when they are needed, these strings are used infrequently.

Syntax for the subscription strings

The following strings relate to subscribing to associations:

- ACF.Subscribe
- ACF.UnSubscribe
- ACF.SubscribeRsp
- ACF.SubscribeMess

The syntax for each string is shown here. Notice that the same response string, ACF.SubscribeRsp, is used for both the ACF.Subscribe and the ACF.UnSubscribe

strings. Zero on the ACF.SubscribeRsp string indicates success; nonzero response values are in Appendix A, "C language header file (ACYAPHDH)," on page 229. For a distinguished name, either the full name or an abbreviated version can be used. The error code 803 indicates that the association does not exist.

```
Subscribe ::= CHOICE {
   ae-title     TitleType,
   association [2] IMPLICIT HandleType
   }

UnSubscribe ::= CHOICE {
   ae-title     TitleType,
   association [2] IMPLICIT HandleType
   }

TitleType ::= CHOICE {
   oi [0] IMPLICIT OBJECT IDENTIFIER
   dn [1] IMPLICIT DistinguishedName
   }

HandleType ::= PrintableString (SIZE(1..36))
```

When the state of an association changes and an application program has registered to receive notification of changes through the ACF.Subscribe string, an ACF.SubscribeMess string is sent to that application program:

```
SubscribeMess ::= SubscribeState
```

The ACF.SubscribeMess syntax does not include the handle of the association whose state has changed. That can be found in the src field of the string header.

In the list of ACF.SubscribeState values, the following values have meaning:

- associated (means the association is established and running)
- terminated (means the association is ended).

The `idle` state is a temporary initial state. The `wait-a-....` states are transitional states. The `wait-a-assoc-...` states indicate that a new association is in the process of being established. The `wait-a-rel-...` states show that an existing association is in the process of being terminated.

```
SubscribeState ::= INTEGER {
   idle                     (0),
   wait-a-assoc-rsp         (1),
   wait-a-assoc-ind         (2),
   wait-a-assoc-cnf         (3),
   wait-a-rel-rsp           (4),
   wait-a-rel-cnf           (5),
   associated               (8),
   wait-a-rel-cnf-indicator (9),
   wait-a-rel-rsp-responder (10),
   terminated               (11)
   }
```

Examples of subscription strings

```
ACF.Subscribe (association 'a2')
ACF.SubscribeRsp 803

ACF.Subscribe (ae-title (dn
   (RelativeDistinguishedName (AttributeValueAssertion
   (attributeType 1.3.18.0.2.4.6, attributeValue NETA)),
   RelativeDistinguishedName (AttributeValueAssertion
   (attributeType 2.9.3.2.7.4, attributeValue (name SSCP1A))),
   RelativeDistinguishedName (AttributeValueAssertion
   (attributeType 1.3.18.0.2.4.12, attributeValue MYAENAME))))
```

```
ACF.SubscribeRsp 0

ACF.UnSubscribe (association s7B1920)
ACF.SubscribeRsp 0

ACF.UnSubscribe (ae-title (dn "1.3.18.0.2.4.6=NETA;2.9.3.2
   .7.4=(name SSCP1A);1.3.18.0.2.4.12=OSISMASE"))
ACF.SubscribeRsp 0
ACF.SubscribeMess  8
```

How the subscription strings are used

To establish a subscription:

1. An application program builds an ACF.Subscribe string and sends it to CMIP services.
2. CMIP services registers the subscription and returns an ACF.SubscribeRsp string to indicate the success or failure of the subscription.
3. When the state of the association changes, CMIP services sends an ACF.SubscribeMess string to the application program containing the new state of the association.

To terminate a subscription:

1. An application program builds an ACF.UnSubscribe string and sends it to CMIP services.
2. CMIP services deletes the subscription and returns an ACF.SubscribeRsp string to indicate the success or failure of the deletion. An ACF.SubscribeRsp string that indicates success does not mean that a subscription did exist.

Registering an application entity

The ACF.RegisterAE request is used to register an explicit application entity with CMIP services. This function can be used if an application program needs to be its own application entity. In general, application programs do not need to use this function. The default local application entity handles all of the application program strings for an association.

An application program must register as its own application entity, if:

- The application program is going to create EFDs.
- The application program needs to request a dedicated association. For a description of how to create a dedicated association, refer to "Creating a dedicated association" on page 140.

Any application program can register an application entity, but only one application program can register any particular application entity. For example, application programs A and B can each register application entities A' and B', but application program B cannot register A' once it has already been registered by application program A.

Any application program can register multiple application entities, but multiple application programs cannot register the same application entity.

Once an application entity has been registered, any associations that are remotely initiated specifying the application entity as the destination of the association are associated directly with the application program that registered the application entity. Any strings that do not include targeting information, such as events, are sent to the application entity directly.

The ACF.RegisterAE request can be used to create an application entity that represents a single application program on CMIP services. This string can be useful if the application program needs to receive event reports directly from other systems.

Syntax of the registration strings

The ACF.RegisterAE request is used to register an application entity.

The syntax for each string is shown here.

```
RegisterAE ::= TitleType

TitleType ::= CHOICE {
   oi [0] IMPLICIT OBJECT IDENTIFIER
   dn [1] IMPLICIT DistinguishedName
   }

RegisterRsp ::= INTEGER {
   success            (0),
   not-accomplished   (1)
   }
```

Examples of RegisterAE strings

The second example, identical to the first, fails because an application entity name can be registered only once by each instance of CMIP services.

```
ACF.RegisterAE (dn "1.3.18.0.2.4.6=NETA;2.9.3.2.7.
   4=(name SSCP1A);1.3.18.0.2.4.12=MYAENAME")
MIB.ServiceAccept()

ACF.RegisterAE (dn "1.3.18.0.2.4.6=NETA;2.9.3.2.7.
   4=(name SSCP1A);1.3.18.0.2.4.12=MYAENAME")
MIB.ServiceError(resultCode 827)
```

How the registration strings are used

To register an application entity title:

1. An application program builds an ACF.RegisterAE request and sends it to CMIP services. CMIP services adds the identification of the source of the string, as with any other string.
2. CMIP services adds the application entity title to the list of supported local application entity titles and sets up communication so that local strings destined for this application entity take the same short path (with no encoding or decoding performed) as the local strings that are sent to the default local application entity.
3. CMIP services associates the name of the instance with the application entity being registered. This information is added to strings that arrive on associations with the application entity by CMIP services.
4. CMIP services responds to the instance indicating that the application entity has been registered.

Starting associations

The ACF.Associate string causes CMIP services to start an association explicitly on behalf of an application program. In general, this string is not needed.

The ACF.Associate string can be used to establish a dedicated association for application programs that require them. For a description of how to create a dedicated association, refer to “Creating a dedicated association” on page 140.

Syntax of the associate strings

The following strings relate to starting an associations:
- ACF.Associate
- ACF.AssociateRsp

The syntax for each string is shown here.

```
Associate ::= SEQUENCE {
   targetAE TitleType,
   securityInfo OCTET STRING OPTIONAL
   }

TitleType ::= CHOICE {
   oi [0] IMPLICIT OBJECT IDENTIFIER
   dn [1] IMPLICIT DistinguishedName
   }
```

Examples of the associate strings

The example that includes the MIB.ServiceError string shows what happens when the target system is not connected: no association can be established.

```
ACF.Associate(targetAE (dn "1.3.18.0.2.4.6=NETA;2.9.3.2.
   7.4=(name SSCP1A);1.3.18.0.2.4.12=MYAENAME"))
ACF.AssociateRsp (handle aF)

ACF.Associate(targetAE (dn "1.3.18.0.2.4.6=NETB;2.9.3.2.
   7.4=(name SSCP1A);1.3.18.0.2.4.12=OSISMASE"))
MIB.ServiceError(resultCode 817)
```

How the associate strings are used

When an application program sends an ACF.Associate string to CMIP services and the application program has already issued the ACF.RegisterAE request, a dedicated association is created. For information about a dedicated association, refer to “Creating a dedicated association” on page 140.

When an application program sends an ACF.Associate string to CMIP services and it has not issued registerAE, a default association is created.

A default association and an association created automatically by CMIP services share the following characteristics:
- Both types of associations can be automatically selected by CMIP services.
- Any application program can destroy the association.
- The association is automatically destroyed by timing out if it is not used.

To establish an association:

1. An application program builds an ACF.Associate string and sends it to CMIP services.
2. CMIP services initiates an association with the desired application entity and returns the newly assigned association handle for the association.

Ending associations

In some cases, an application program knows that an association should be ended. The ACF.Release and ACF.Abort strings indicate that an association should be ended gracefully (ACF.Release) or abruptly (ACF.Abort). The ACF.Release string ensures that all pending messages have cleared before the association is ended.

If the association is ended successfully, the MIB.ServiceAccept string is sent. If the association is not ended successfully, the MIB.ServiceError string is sent. For description of these strings, refer to "Requests and responses with the MIB prefix" on page 141.

Syntax of the ACF.Release and ACF.Abort strings

The following strings relate to ending associations:
- ACF.Release
- ACF.Abort

The syntax for each string is shown here.

```
Release ::= SEQUENCE {HandleType}

HandleType ::= PrintableString (SIZE(1..36))

Abort ::= SEQUENCE {HandleType}
```

Examples of the ACF.Release and ACF.Abort strings

Note that the example that includes the MIB.ServiceError string has an extra right parenthesis.

```
ACF.Release (a4))
MIB.ServiceError(resultCode 345,resultMessage "msg ACF.Release (a4 ) )"

ACF.Abort (a8)
MIB.ServiceAccept()

ACF.Release (s17B1440)
MIB.ServiceAccept()
```

How the ACF.Release and ACF.Abort strings are used

An application program sends either an ACF.Release or ACF.Abort string containing the identification of the association to be ended. If the association exists, it is ended. CMIP services sends the ACF.AssociateRsp string to the application program.

Getting association information

In some cases an application program needs to learn about an active association. An application program can request a number of items corresponding to a specific association. CMIP services returns values for the following attributes:
- state
- partner-AE-title
- securityInfo
- peerAuthenticationPerformed

Syntax of the GetAssociationInfo string

The ACF.GetAssociationInfo string gathers information about an active association.

This syntax for each string is shown here.

```
GetAssociationInfo ::= SEQUENCE {
   handle  GraphicString,
   info    BIT STRING {
              state                        (0),
              assoc-handle                 (1),
              sess-handle                  (2),
              partner-AE-Title             (3),
```

```
                application-context           (4),
                presentation-context-def-list (5),
                securityInfo                  (6),
                peerAuthenticationPerformed   (7),
                }
   }

AssociationInfo ::= SET OF InformationPair

InformationPair ::= SEQUENCE {
  label GraphicString,
  value GraphicString
  }
```

Examples of the GetAssociationInfo string

The first example includes the MIB.ServiceError string because the message did not specify as many zeros or ones as there are bits in the bit string.

The remaining examples show successful use of ACF.GetAssociationInfo.

```
ACF.GetAssociationInfo(handle 'a1', info 00010)
MIB.ServiceError(resultCode 804)

ACF.GetAssociationInfo(handle a1, info 00000000)
ACF.AssociationInfo ()

ACF.GetAssociationInfo(handle 'a2', info 00010000)
ACF.AssociationInfo ((partner-AE-Title '1.3.18.0.2.4.6=N
    ETA;2.9.3.2.7.4=(name "SSCP1A");1.3.18.0.2.4.12=OSISMASE'))

ACF.GetAssociationInfo(handle 'a3', info 10010000)
ACF.AssociationInfo ((state 8),(partner-AE-Title '1.3.18
    .0.2.4.6=NETA;2.9.3.2.7.4=(name "SSCP1A");1.3.18.0.2.4.12=OSISMASE'))

ACF.GetAssociationInfo(handle 's147B290', info 10010011)
ACF.AssociationInfo ((state 8),(partner-AE-Title '1.3.18
    .0.2.4.6=NETA;2.9.3.2.7.4=(name "SSCP2A");1.3.18.0.2.4.12=OS
    ISMASE'),(securityInfo ""),(peerAuthenticationPerformed TRUE))

ACF.GetAssociationInfo(handle 'aA', info 11111111)
ACF.AssociationInfo ((state 8),(partner-AE-Title '1.3.18
    .0.2.4.6=NETA;2.9.3.2.7.4=(name "SSCP2A");1.3.18.0.2.4.12=OS
    ISMASE'),(securityInfo A1B2C3D4),(peerAuthenticationPerformed FALSE))
```

How the GetAssociationInfo string is used

An application program sends a GetAssociationInfo string to CMIP services, filling in the types of information it requires. CMIP services returns an AssociationInfo string containing the desired information.

The labels used to identify the information on the response are identical to the named bits on the request.

The value is the value corresponding to the label. For `securityInfo`, the value is the information passed (if any) from the association partner when the association was requested. For securityInfo, the value is saved only on the target CMIP services.

For `peerAuthenticationPerformed`, the value (on both initiating and target systems) is 0 if no authentication is performed by CMIP services and 1 if DES-based security is performed for this association by CMIP services.

Creating a dedicated association

A dedicated association is restricted as to who can use it on the CMIP services that created the association. A dedicated association has the following characteristics:

- It is only used if specifically requested by the application program that sends the ACF.Associate string to CMIP services.
- It can only be destroyed by an ACF.Release or ACF.Abort string from the application program that sent the ACF.Associate string.

Note: On the other CMIP services, the association is not flagged as dedicated. Therefore, it can time out or be used by any application program.

In some cases, application programs need to monitor the existence of remote systems. For example, an application program might need to be aware when a remote system fails. Having EFDs on that remote system helps only in cases when actual communication remains intact. If connectivity to the remote system is lost, the application program might not be notified of the event. If the application program needs to know that connectivity is lost, the application program can start a dedicated association to the remote system and monitor it for failures.

Idle CMIP associations are terminated by CMIP services on a regular basis, according to a timer:

- If limited resources is enabled, the limited resources timer is used.
- If limited resources is not enabled, the CMIP services timer is used. The CMIP services timer terminates idle associations every 2 hours.

Shared associations, which are those started automatically by CMIP services on an as-needed basis, are terminated when the timer expires, unless the association is being used for an outstanding CMIP operation.

Dedicated associations are not terminated on the originating system even if there is no outstanding work. Note that remote systems, which are those that did not initiate the dedicated association, are not aware that the association is dedicated. The remote systems treat the association as shared. The remote systems terminate the idle association when the timer on the remote system expires.

To prevent associations from being automatically terminated, you can maintain a never-ending operation on the association. For example, one application program can be designed to have a special object that never responds to a particular operation. Another application program can then issue this special operation to that object, solely for the purpose of maintaining a never-ending operation on the association.

The application programs can continue to send or receive other operations on that same association.

In addition to ensuring that the association remains active, an application program can monitor an association by subscribing to it. When an application program subscribes to an association, the application program is notified if the association is terminated. For a description of how to subscribe to an association, refer to “Subscribing to association information” on page 133.

To create a dedicated association, an application program must do the following:

- Register an application entity (AE) title. Refer to “Registering an application entity” on page 135 for more information.

- Establish an association with the remote system as the target application entity. Refer to "Starting associations" on page 136 for more information.
- Subscribe to the association. Refer to "Subscribing to association information" on page 133 for more information.

Requests and responses with the MIB prefix

The following requests and responses are described in this section:
- MIB.GeneralRequest
- MIB.GeneralResponse
- MIB.GeneralError
- MIB.ServiceError
- MIB.ServiceAccept
- MIB.RegisterAccept

MIB.GeneralRequest, MIB.GeneralResponse, and MIB.GeneralError

These messages are built on behalf of the application program by the MIBSendCmipRequest and MIBSendCmipResponse functions. An application program does not build them and an application program will not receive them. They appear in buffer traces of application programs that call the MIBSendCmipRequest or MIBSendCmipResponse functions.

MIB.ServiceError

The MIB.ServiceError message is sent to an application program from CMIP services when a request or response from the application program cannot be processed for some reason. Some example reasons are parsing errors in the request, network errors trying to reach the destination object, or memory allocation errors.

For some types of errors, additional information will be provided in the optional resultMessage section of the ServiceError SEQUENCE.

Here is a sample ServiceError as received by an application, including the APIhdr:

```
03000000 00030017 00000003 00000001    *................*
2FAA7B32 013D0000 00000000 00000001    *..#............*
94A28740 D4C9C24B E28599A5 898385C5    *msg MIB.ServiceE*
99999699 4D9985A2 A493A3C3 96848540    *rror(resultCode *
F3F1F76B 9985A2A4 93A3D485 A2A28187    *317,resultMessag*
85407FA4 948595A3 404DF24B F94BF34B    *e "ument (2.9.3.*
F24BF34B F1F36BC0 91D08586 86D48195    *2.3.13,.j.effMan*
81878584 D6829185 83A3C995 A27F5D00    *agedObjectIns").*
```

The position of the string where parsing stopped is delimited in the portion of the original message byX'C0' and X'D0'. In this case, the character pointed out is *j* of jeffManagedObjectInstance. This label should instead be baseManagedObjectInstance.

MIB.ServiceAccept

The MIB.ServiceAccept message is sent to an application program from CMIP services when the application program sends an unconfirmed CMIP request or a CMIP response. Its purpose is to notify the application program that the request or response was processed correctly.

Here is a sample MIB.ServiceAccept as received by an application program including the APIhdr:

```
02000100 00030011 00000001 00000001    *................*
2FAA54E5 00000000 00000000 00000001    *...V............*
A2998360 A3A89785 40F16B40 A2998340    *src-type 1, src *
81F16B40 94A28740 D4C9C24B E28599A5    *a1, msg MIB.Serv*
898385C1 83838597 A34D5D00             *iceAccept().    *
```

MIB.RegisterAccept

The MIB.RegisterAccept message is sent to an application program from CMIP services when an object is successfully registered by that application program.

An object can be successfully registered even if one or more items in the allomorphs list or create handler list cannot be processed. In this case, information about allomorphs or create handler failures will be in the MIB.RegisterAccept message.

Here is an example MIB.RegisterAccept as received by an application program including the APIhdr:

```
                  01000000 00030018    *        ........*
00000003 00000001 2FAA7CF4 00000000    *.........@4....*
00000000 00010000 A2998360 A3A89785    *........src-type*
40F16B40 A2998340 81F16B40 94A28740    * 1, src a1, msg *
D4C9C24B D9858789 A2A38599 C1838385    *MIB.RegisterAcce*
97A34D95 819485C2 89958489 958740F1    *pt(nameBinding 1*
4BF34BF1 F84BF04B F04BF2F1 F7F26B40    *.3.18.0.0.2172, *
81939396 94969997 88A2C599 999699D3    *allomorphsErrorL*
89A2A340 4D5D6B40 83998581 A385C881    *ist (), createHa*
95849385 99C59999 9699D389 A2A34D5D    *ndlerErrorList()*
5D00                                   *).              *
```

Chapter 11. Application-program-to-application-program security

In VTAM CMIP services, there are two kinds of security:
- System-to-system security
- Application-program-to-application-program security.

System-to-system security is between two instances of CMIP services or between CMIP services and itself. See *z/OS Communications Server: SNA Network Implementation Guide* and *z/OS Communications Server: SNA Resource Definition Reference* for more information about this type of security.

This chapter describes how to define application-program-to-application-program security.

For any particular association, application-program-to-application-program security and DES-based system-to-system security are mutually exclusive. If two application programs decide to implement application-program-to-application-program security and the CMIP Services for each application program has defined DES-based security to be used between the two CMIP services, the association between the two application programs fails. To establish a secure association between two application programs on two instances of CMIP services, choose one of the following methods:
- Use DES-based security for all associations between the two instances
- Use application-program-to-application-program security for all associations between the two instances
- Register an application entity title for one or both application programs. Use application-program-to-application-program security between the two application entities. For a description of how to register an application entity title, refer to "Registering an application entity" on page 135.

To use application-program-to-application-program security, you need to understand the sequence of strings sent between each instance of CMIP services and the application programs attempting to set up an association.

For syntax and details about the particular strings, refer to Chapter 10, "VTAM-specific requests and responses," on page 133.

If the two instances of CMIP services for the application programs have defined the associationKey attribute `associationKey '.'` in the directory definition files, follow the steps here to specify application-program-to-application-program security.

As you read the steps, refer to Figure 5 on page 144 for an illustration. The numbers in the figure correspond to the steps.

The steps refer to an *origin application program* and a *target application program*. The origin application program requests that the association be established with the target application program. For example, the origin application program might be a manager application program, and the target application program might be an agent application program.

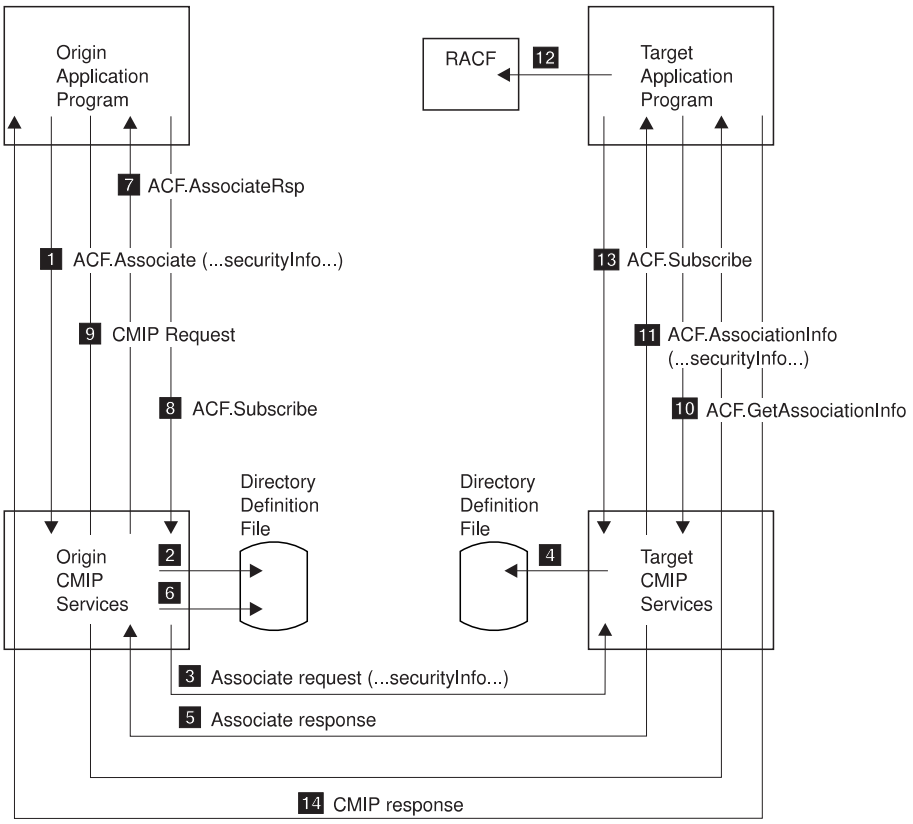


Figure 5. Application-program-to-application-program security

1. The origin application program decides to communicate with a target application program. The origin application program issues an ACF.Associate string to its CMIP services. The ACF.Associate string includes the securityInfo attribute.
2. The origin application program checks the directory definition file. The directory definition file indicates that the association is allowed to be established between the two application programs. The file also indicates that the securityInfo attribute must be passed to the target application program.
3. CMIP services sends the securityInfo value on the associate request to the other CMIP services.
4. If the receiving CMIP services is not using DES-based security, that CMIP services discovers the securityInfo value. CMIP services assumes that the securityInfo information is tied to that particular association.
5. The CMIP services for the target application program sends a positive response to the associate request.
6. The CMIP services for the origin application program checks the directory definition file again, in case it has changed.
7. When the association between the two CMIP services has been established successfully, CMIP services returns an association handle, which identifies to the application program this particular association.
8. The origin application program subscribes to that association so that the origin application program can be informed when the association ends.

   So long as the CMIP services association stays intact, the partner at the other end is the same on subsequent requests as it was on the initial request.

However, associations can be terminated independently of either the origin application program or the target application program. For example, either of the following could end an association:

- The limited resources function in VTAM that allows the selective termination of idle LU 6.2 (APPC) sessions can end an association by terminating a session that the association is using.
- CMIP services can end an association that has not been used.

For more information about the effects of the VTAM limited resources function and the CMIP services automatic termination of associations, refer to "Starting associations" on page 136.

Because associations can be reused for associations between different partners at a later time, the application programs on both ends of an association need to be aware of the association status.

In this example, the origin application program initiates the association. If the association goes down, the origin application program needs to initiate another association and possibly reissue requests that are outstanding at the time the prior association ended. (CMIP services for the origin application program returns an error when requests are sent over an association that has ended.)

To be aware of association termination, the origin application program can either issue an ACF.Subscribe string for that association or wait until an error code is returned from the MIBSendCmipRequest or MIBSendRequest function.

9. The origin application program sends to the target application program a CMIP request. The origin application program identifies the appropriate association by using the association handle that was returned in step 7 on page 144.

   CMIP services routes the request over only the designated association. This is the first time that the target application program is aware of the existence of the origin application program.

   If CMIP services cannot route the request over that association, CMIP services returns the string `MIB.ServiceError (resultCode nnn)` to the application program, where *nnn* is the number of the error code.

   For the names of each error code number, refer to the language header file, ACYAPHDH, or Appendix A, "C language header file (ACYAPHDH)," on page 229. For the names with descriptions of each error code number, refer to Appendix C, "Error codes sent by CMIP services," on page 263.

10. If this is the first CMIP request issued by the origin application program after requesting the association, the target application program has no prior knowledge of the particular association. The target application program therefore issues the GetAssociationInfo string to CMIP services, specifying the particular association handle.
11. CMIP services returns the requested information on the ACF.AssociationInfo string and includes the securityInfo value that was obtained by the origin CMIP services.
12. On *every* CMIP request, the target application program is required to verify whether the requesting application program has authority to issue such a request and to receive a valid response. In this example, the target application program accesses a security function, such as Resource Access Control Facility (RACF®).
13. As mentioned in step 8 on page 144, the application programs on both ends of the association need to be aware of the association status. Therefore, the target application program should also issue the ACF.Subscribe string to its CMIP

services. If the target application program ever receives an ACF.SubscribeMess string indicating that the association is no longer active, the target application program should discard its knowledge of this association, since this information is no longer valid.

14. The target application program responds to the CMIP request.

Note that most of these steps occur only when the association is being established. Once the association is established, only steps 9, 12, and 14 are performed.

Part 2. VTAM topology agent

Chapter 12. Introduction to VTAM topology agent

In VTAM, CMIP services is made available by specifying the OSIMGMT=YES start option. This start option gives you access to the following:
- VTAM topology agent
- CMIP services

The VTAM topology agent is a part of VTAM that functions as a CMIP application program. It is designed to communicate through the application program interface that is part of VTAM CMIP services with a network manager application program, such as the NetView program. For information on the manager function, refer to *TME 10™ NetView for OS/390 SNA Topology Manager and APPN Accounting Manager Implementation Guide*.

The information provided by the VTAM topology agent and CMIP services allows users at a topology manager application program to monitor resource status and to manage the network. The manager application program is installed, started, and maintained separately from VTAM.

The basic function of the VTAM topology agent is to provide the capability for monitoring the topology of a VTAM network. The VTAM topology agent provides this capability by supplying the following topology information:
- Local topology
- Network topology
- LUs that VTAM owns
- LUs that are owned by another node but are known to this VTAM

The VTAM topology agent supplies the topology information by:
- Responding to requests for data
- Providing unsolicited data

The following sections give the details of the CMIP operations that the VTAM topology agent supports and describe the data supplied by the topology agent for those CMIP operations. Chapter 13, “OSI object classes and VTAM resources,” on page 151 describes how the VTAM topology agent maps VTAM resources to OSI objects.

Chapter 14, “OSI operations,” on page 159 discusses the OSI operations that are performed on the objects, the CMIP responses and errors that the topology agent provides, and the general resource monitoring process.

Chapter 15, “VTAM topology monitoring,” on page 171 describes the specific resource-monitoring operations. This chapter describes how to request the monitoring and explains the data VTAM provides.

Chapter 16, “Requesting specific resource data,” on page 219 describes how the VTAM topology agent gathers information about specific resources.

For reference, the following lists are included in the appendixes:
- Appendix E, “VTAM topology agent object and attribute tables,” on page 301 contains a list of all object classes supported by the VTAM topology agent, the operations that are supported for each class, and a list of the supported attributes for each object class.

- Appendix F, “VTAM topology agent attributes definition,” on page 313 contains a comprehensive list of all supported attributes, including a description of the semantics of the attribute, the syntax of the attribute, and the uses of the attribute.

Chapter 13. OSI object classes and VTAM resources

This chapter describes of the OSI object definitions supported by the VTAM topology agent.

OSI object classes

The VTAM topology agent provides topology data for the network resources that VTAM manages. To do this the VTAM resources must first be viewed in the object-oriented context of the OSI object definitions. The following list shows the OSI object classes supported by the VTAM topology agent:

- crossDomainResource
- crossDomainResourceManager (valid only in replies, not requests)
- definitionGroup
- appnEN
- interchangeNode
- lenNode
- logicalLink
- logicalUnit
- luCollection
- luGroup
- logicalUnitIndex
- migrationDataHost
- appnNN
- port
- appnRegisteredLu
- snaLocalTopo
- snaNetwork
- appnTransmissionGroup (valid only in replies, not requests)
- subareaTransmissionGroup (valid only in replies, not requests)
- t2-1Node
- t4Node
- t5Node
- virtualRoute (valid only in replies, not requests)
- virtualRoutingNode (valid only in replies, not requests)

Each object class definition contains a list of attributes for that class. The attributes supported by VTAM are listed by object class in Appendix E, "VTAM topology agent object and attribute tables," on page 301. Only the object classes that are valid in requests are listed.

Although some of the object classes have obvious meanings, some represent resources in VTAM that are known by different names. The next sections address the mapping of VTAM resources back to these OSI object classes. Note that some of the OSI classes do not represent existing VTAM objects; these OSI objects generally represent a group of VTAM resources. These object types are described more fully in Chapter 15, "VTAM topology monitoring," on page 171.

Mapping VTAM resources to OSI object classes

The resources that VTAM manages are known traditionally by a somewhat different set of names than the OSI object classes. For the VTAM resources with different names, the following table shows the mapping to the OSI classes:

Table 8. VTAM resources mapped to OSI classes

VTAM resource	OSI class
physical unit	logicalLink
linkstation	logicalLink
application	logicalUnit
dependent LU	logicalUnit
independent LU	crossDomainResource
CDRSC	crossDomainResource
CDRM	crossDomainResourceManager
type 5 node	t5Node
type 4 node (NCP)	t4Node
type 2.1 node	t2-1Node
APPN end node	appnEN
APPN network node	appnNN
interchange node	interchangeNode
migration data host	migrationDataHost
line	port
channel	port
appnRegisteredLu	appnRegisteredLu
subarea TG	subareaTransmissionGroup
APPN TG	appnTransmissionGroup
major node	definitionGroup
USERVAR	luGroup
generic resource	luGroup

Naming the objects

Each instance of an object class is called an object instance. Because an object instance consists only of attributes and behavior, there is not an object instance name assigned to the instance. As in many object-oriented systems, one attribute is assigned to contain a value that is used to name the object instance. This attribute is called the *naming attribute*. Unlike some other systems, instance names in the VTAM topology agent do not consist solely of the value of the naming attribute. Instead, instances are identified by their *distinguished names* (DNs). The distinguished name consists of a sequence of *relative distinguished names* (RDNs), each of which contains an attribute value assertion (AVA).

Consider this example: we have a network, NETA, and a VTAM type 5 node, SSCP1A, and a channel attached to SSCP1A called 0321-L. The channel is considered a *port* object. It has a distinguished name that is composed of three relative distinguished names. In Figure 6 on page 153, the leftmost name is the first

relative distinguished name, the next name is the second, and the rightmost name is the third.

```
snaNetId=NETA ; snaNodeName=SSCP1A ; portId=0321-L
Where:
 - snaNetId=NETA is the first relative distinguished name.
 - snaNodeName=SSCP1A is the second relative distinguished name.
 - portID=0321-L is the third relative distinguished name.
```

Figure 6. Distinguished name composed of three relative distinguished names

The name of the port object in this example suggests a hierarchy. The leftmost relative distinguished name is the highest level in the hierarchy, the network identifier. The next name is lower in the hierarchy, the node name. And the rightmost name is the lowest in the hierarchy, the resource name. This naming convention is called *name-containment*.

The port object name must be unique within the domain of the SSCP. The name is made unique among all SSCPs in the network by qualifying it with the SSCP name. The name is made universally unique by qualifying it with the network ID as well, assuming the network ID is unique.

VTAM-managed objects are named under VTAM, which is a *snaNode* object with the naming attribute *snaNodeName*. VTAM is named under a *netIDsubnetwork* object, with the naming attribute *snaNetId*. The distinguished names are actually a sequence of naming attributes and their values, starting at the highest level object and moving toward the lowest level object.

A summary of the distinguished names for the VTAM-managed objects is given in Table 9.

Table 9. Object names and shorthand distinguished names

Object name	Shorthand distinguished names
t2-1Node	*snaNetID*=netid; *snaNodeName*=CPname
lenNode	*snaNetID*=netid; *snaNodeName*=CPname
appnNN	*snaNetID*=netid; *snaNodeName*=CPname
appnEN	*snaNetID*=netid; *snaNodeName*=CPname
virtualRoutingNode (See note 1.)	*snaNetID*=netid; *virtualRoutingNodeName*=CPname (of the virtual routing node)
interchangeNode	*snaNetID*=netid; *snaNodeName*=CPname
migrationDataHost	*snaNetID*=netid; *snaNodeName*=CPname
t5Node	*snaNetID*=netid; *snaNodeName*=SSCPname
t4Node	*snaNetID*=netid; *snaNodeName*=SSCPname; *snaNodeName*=NCPname
logicalUnit	One of the following: • *snaNetID*=netid; *snaNodeName*=CPname; *luName*=netid.LUname • *snaNetID*=netid; *snaNodeName*=CPname; *luName*=LUname
crossDomainResourceManager (See note 2.)	One of the following: • *snaNetID*=netid; *snaNodeName*=CPname; *snaNodeName*=netid.CDRMname • *snaNetID*=netid; *snaNodeName*=CPname; *snaNodeName*=CDRMname
crossDomainResource (See note 3.)	One of the following: • *snaNetID*=netid; *snaNodeName*=CPname; *nonLocalResourceName*=netid.CDRSCname • *snaNetID*=netid; *snaNodeName*=CPname; *nonLocalResourceName*=CDRSCname

Table 9. Object names and shorthand distinguished names (continued)

Object name	Shorthand distinguished names
appnRegisteredLu	One of the following: • *snaNetID*=netid; *snaNodeName*=CPname; *nonLocalResourceName*=netid.regLUname • *snaNetID*=netid; *snaNodeName*=CPname; *nonLocalResourceName*=regLUname
logicalUnitIndex	One of the following: • *snaNetID*=netid; *snaNodeName*=CPname; *logicalUnitIndexName*=netid.LUname • *snaNetID*=netid; *snaNodeName*=CPname; *logicalUnitIndexName*=LUname
luGroup	*snaNetID*=netid; *snaNodeName*=CPname; *luGroupName*=USERVAR or generic resource name
luCollection	One of the following: • *snaNetID*=netid; *snaNodeName*=CPname; *luCollectionId*=luCollection • *snaNetID*=netid; *snaNodeName*=CPname; *linkName*=PUname; *luCollectionId*=luCollection
port	*snaNetID*=netid; *snaNodeName*=CPname; *portId* = LINEname
logicalLink	One of the following: • *snaNetID*=netid; *snaNodeName*=CPname; *linkName*=linkstation_name or PUname • *snaNetID*=netid; *snaNodeName*=CPname; *linkName*=netid.linkstation_name or netid.PUname
virtualRoute	*snaNetID*=netid; *snaNodeName*=CPname; *virtualRouteId*= netid.originSubareaNumber.destSubareaNumber.virtualRouteNumber. transmissionPriority
appnTransmissionGroup	*snaNetID*=netid; *snaNodeName*=CPname; *transmissionGroupId*=TGN.partner_NETID.partner_CPNAME
subareaTransmissionGroup (See note 4.)	One of the following: • *snaNetID*=netid; *snaNodeName*=CPname; *transmissionGroupId*=local_subarea.TGN.partner_NETID. partner_subarea.partner_node • *snaNetID*=netid; *snaNodeName*=CPname; *snaNodeName*=NCPname; *transmissionGroupId*=local_subarea.TGN.partner_NETID. partner_subarea.partner_node
definitionGroup	*snaNetID*=netid; *snaNodeName*=CPname; *definitionGroupName*=mjnode_type.mjnode_name
snaNetwork	*snaNetID*=netid; *snaNodeName*=CPname; *graphid*=(string "SnaNetwork")
snaLocalTopo	*snaNetID*=netid; *snaNodeName*=CPname; *graphid*=(string "SnaLocalTopology")

Notes:

1. This object is identified by a role attribute, which is a pointer that consists of the distinguished name of the object, but the object actually does not reside on the local node.
2. A CDRM name is not required to match the real SSCP name of the type 5 node it represents. CDRMs might be defined such that the network identifier is known and unchangeable. Therefore, CDRM objects are named using the CDRM name defined to VTAM. A CDRM with a predefined network identifier includes the network identifier in the last RDN™. A CDRM without a predefined network identifier does not include the network identifier in the last RDN. The real name and network identifier of the adjacent SSCP are available in the *realSSCPname* attribute when the CDRM is active.
3. A CDRSC that represents a predefined alias does not include a network identifier in the last RDN because the network identifier can change. A CDRSC with a predefined network identifier includes the network identifier in the last RDN. The CDRSC name used in this RDN is the name that was predefined for

the CDRSC. This name is not necessarily the same as the real name of the resource that the CDRSC maps. The real name and network identifier of the resource is provided in the *cdrscRealLUname* attribute of the *crossDomainResource* object.

4. Partner_node name for subarea transmission groups can be formed from the subarea number if the contacted subarea node does not provide its name in the X'0E' control vector on contacted.

OSI object states

Among the many attributes contained within the managed objects, some of the most important attributes are the *state attributes*. The state attributes consist of the six attributes defining the OSI states, as documented in the ISO/IEC 10164-2 standard, in addition to a seventh state, which represents the normal VTAM resource status.

Each OSI state attribute is described in the following list:

- **Operational state attribute**

  Indicates whether a resource is operational, according to the following values:

  Enabled
  The resource is partially or fully operational and available for use.

  Disabled
  The resource is not partially or fully operational and is not available for use.

- **Usage state attribute**

  Indicates whether a resource is in use, according to the following values:

  Idle The resource is not in use.

  Active The resource is in use and has sufficient spare operating capacity to provide for additional users simultaneously.

  Busy The resource is in use and it has no spare operating capacity to provide for additional users at this instance.

- **Administrative state attribute**

  Indicates whether a resource is allowed to perform functions. The administrative state of a managed object is determined separately from the operational and usage states. Administrative state can have the following values:

  Unlocked
  The resource is permitted to perform services for its users.

  Locked
  The resource is prohibited from performing services for its users.

  Shut down
  Only existing instances are permitted to use the resource.

- **Procedural status attribute**

  This attribute is supported by only those classes of managed objects that represent some procedure (for example, a test process) that progresses through a sequence of phases. The procedural status attribute can have the following values:

  Not initialized
  Indicates that the resource must be initialized before it can perform normal functions. The initialization procedure has not been started.

Initializing
Indicates that the resource must be initialized before it can perform normal functions. The initialization procedure has been started, but is not yet complete.

Terminating
Indicates that this resource is in a termination phase.

- **Availability status attribute**

  Offline
  Indicates that the resource requires a routine operation to place it online and make it available for use.

  Intest Indicates the resource is undergoing a test procedure.

  Degraded
  Indicates overuse of cycles or buffers.

  Dependency
  Indicates that a higher-level resource is in a state of transition, either up or down. For related information on dependency, refer to "ACTION(snapshot) update merging" on page 168.

  Failed Indicates that a resource is inoperative.

- **Unknown status attribute**

  Indicates that the state of the resource represented by the managed object is unknown. When the unknown status attribute value is true, the value of the state attribute cannot reflect the actual state of the resource.

- **Native status attribute**

  Indicates the VTAM internal state of the resource.

Mapping VTAM status to OSI states

The topology agent maps the existing status of VTAM resources to OSI states when the topology agent reports object data. For traditional subarea resources, the mapping is straightforward; however, for some APPN resources, a VTAM status does not exist.

Table 10 shows how VTAM resource status is mapped to OSI states. Table 11 on page 158 shows the valid combinations of OSI states for the resources with no applicable VTAM status.

OSI states for VTAM resources with VTAM status

In Table 10, to find the OSI state for a VTAM resource with a particular VTAM status, find the status in the first column. Follow the row across until it intersects with the column for your VTAM resource. The abbreviation in that cell of the table indicates the OSI state that is assigned. Look up that abbreviation in the list of abbreviations at the end of the table.

Table 10. VTAM resource status to OSI atates

VTAM resource status	Native status	NCP	CDRM	LU and CDRSC	Type 2 PU and type 2.1 PU	Link station	Line	VTAM agent host
ACT	ACT	uea	uea	uei	uea	uea	uea	uea
ACT/S	ACT/S	N/A	N/A	uea	N/A	N/A	N/A	N/A
INACT	INACT	udi	udi	udi	udi	udi	udi	N/A

Table 10. VTAM resource status to OSI atates (continued)

VTAM resource status	Native status	NCP	CDRM	LU and CDRSC	Type 2 PU and type 2.1 PU	Link station	Line	VTAM agent host
INACTIVE (INOP) (See note 1)	INACTIVE	udi-fl	udi-fl	udi-fl (for LUs) udi (for CDRSCs)	udi-fl	udi-fl	udi-fl	N/A
INACTIVE (NEVAC)	NEVAC	udi-ni	udi-ni	udi-ni	udi-ni	udi-ni	udi-ni	N/A
PND-ACT	PND-ACT	udi-in	udi-in	udi-in	udi-in	udi-in	udi-in	N/A
PND-INACT	PND-INACT	uea-tm	uea-tm	uea-tm	uea-tm	uea-tm	uea-tm	N/A
CONNECT-ABLE	CONNECT-ABLE	N/A	N/A	uei-ol (Switched resources only)	uei-ol (Switched resources only)	N/A	N/A	N/A
ROUTABLE	ROUTABLE	N/A	N/A	uei-it	uei-it	N/A	N/A	N/A
ROUTABLE (released)	ROUTABLE	N/A	N/A	uei-it-ol	uei-it-ol	N/A	N/A	N/A
ACTIVE (congested)	ACTIVE	ueb	N/A	N/A	ueb	N/A	N/A	N/A
ACT	DISABLE	N/A	N/A	uei-po	N/A	N/A	N/A	N/A
INACTIVE (released)	RELEASED	udi-ol	N/A	udi-ol	udi-ol	udi-ol	udi-ol	N/A
INACTIVE (reset)	RESET	udi-unkwn	udi-unkwn	udi-unkwn	udi-unkwn	udi-unkwn	udi-unkwn	N/A
PND-INACT (reset)	PND-INACT	uea-tm-unkwn	uea-tm-unkwn	uea-tm- unkwn	uea-tm- unkwn	uea-tm-unkwn	uea-tm-unkwn	N/A

Note:

The OSI states and statuses are from the ISO/IEC 10164-2 standard. The states are listed in the following order:

- Administrative state (administrativeState)
- Operational state (operationalState)
- Usage state (usageState).

For a description of the states, refer to "OSI object states" on page 155.

OSI status	Description
uea	Unlocked Enabled Active
ueb	Unlocked Enabled Busy
uei	Unlocked Enabled Idle
udi	Unlocked Disabled Idle
-tm	proceduralStatus = terminating
-ol	availabilityStatus = offline
-fl	availabilityStatus = failed
-it	availabilityStatus = in test
-po	availabilityStatus = power off
-in	proceduralStatus = initializing
-ni	proceduralStatus = not initialized
-unkwn	unknownStatus = unknown
N/A	Not Applicable

Footnotes for the table entries:

1. A resource is considered INOP when the VTAM display of the resource shows INOP. In most cases, the internal VTAM status of an INOP resource is INACTIVE. However, it is possible to have other status values also showing INOP. Therefore, the availabilityStatus value `failed` might appear with nativeStatus values other than INACTIVE.

2. An additional value of `dependency` might be added to the *availabilityStatus* attribute if a higher-level resouce is in transition.
3. NCP slow down indication always forces usage state=busy.

OSI states for VTAM resources without VTAM native status

For information about the status of APPN network nodes and transmission groups, refer to Table 11. Note that the *nativeStatus* attribute does not apply to these resources.

Table 11. OSI states for VTAM resources without native status

VTAM resource	OSI state
appnTG	uea uea-tm udi udi-ol
networkNode or interchange node (As reported in snaNetwork APPN network topology.)	uea uea-tm ueb ueb-tm uea-dg ueb-dg udi-ol

Description of the OSI resource statuses:

The OSI states and statuses are from the ISO/IEC 10164-2 standard. The states are listed in the following order:

- Administrative state (administrativeState)
- Operational state (operationalState)
- Usage state (usageState)

For a description of the states, refer to "OSI object states" on page 155.

OSI Status

	Description
uea	Unlocked Enabled Active
ueb	Unlocked Enabled Idle
udi	Unlocked Disabled Idle
-dg	availabilityStatus = degraded
-tm	proceduralStatus = terminating
-ol	APPN garbage collection indicator

Chapter 14. OSI operations

This chapter describes the OSI operations that are performed on the objects that were described under Chapter 13, "OSI object classes and VTAM resources," on page 151. The following topics are included:

- Introduction to the CMIP verbs that are used to specify the operations and overview of the VTAM topology agent processing of the operations
- Overview of types of responses the VTAM topology agent provides to an input CMIP request
- High-level description of the resource-monitoring process using an ACTION(*snapshot*) operation

The details of the specific monitoring capabilities of the VTAM topology agent are provided under Chapter 15, "VTAM topology monitoring," on page 171.

Specifying OSI operations with CMIP verbs

The set of OSI operations, specified by CMIP verbs, is used to collect topology data about VTAM resources represented as object instances.

Not all object classes are supported for all operations; in some cases, object classes are supported only for response data and not for request data.

Performing an operation on an object instance usually involves a manager application program sending a CMIP request message (ROIV) to the object. The message contains an indication of the type of operation being requested, as well as other data related to the requested operation. The VTAM topology agent receives the request message, performs the requested operation, and generates and sends a CMIP response message to the manager application program. This sequence is altered slightly when objects send unsolicited messages or requests that provide information about an **event** that has occurred.

A given operation is either *confirmed* or *unconfirmed*. A confirmed operation is one that requires that a response be returned to the application program that issued the request. An unconfirmed operation is one for which there can be no response.

The following operations, supported by the VTAM topology agent, are described in more detail in the following sections:

- GET
- CANCEL-GET
- ACTION
- EVENT-REPORT
- SET
- DELETE
- Other operations

GET

GET is a confirmed request that is issued by a manager application program. The request is directed to an object instance, requesting the return of attribute data for that object instance. The GET response contains the requested attribute data.

CANCEL-GET

CANCEL-GET is a confirmed request that is issued by a manager application program. The function of this operation is to terminate the processing of a GET request previously issued by this manager application program. The CANCEL-GET response message contains only an indication of whether the GET request was successfully terminated.

ACTION

The ACTION operation has two types: confirmed and unconfirmed. The types are usually specified as:

- ACTION, which is unconfirmed
- ACTION-CONFIRMED, which is confirmed

This discussion refers only to ACTION-CONFIRMED. ACTION is a multi-function operation. The ACTION request is requesting the target object to do one of a set of possible functions. The particular function being requested is specified by the detailed **actionType** data contained in the request. The VTAM topology agent supports only one type of action, ACTION(*snapshot*). The ACTION(*snapshot*). operation is a request directed to one of a set of special objects, requesting the return of a set of topology data. The type and amount of data returned varies, depending upon the class of object that is the target of the request.

SET

The SET operation has both confirmed and unconfirmed types. The SET request is directed to an object instance, requesting that specified attributes for that object be set to values provided in the SET request. The VTAM topology agent does not support the setting of VTAM resource data by using the CMIP SET operation. However, the VTAM topology agent does respond to any confirmed SET request it receives. For a discussion of error responses, refer to "Responding to CMIP requests" on page 161.

DELETE

DELETE is a confirmed operation directed to an object instance. The intended function of the DELETE operation is to request that an object instance be deleted. The DELETE response contains an indication of whether the object was actually deleted. The VTAM topology agent does not support the deletion of VTAM resources by using the CMIP DELETE operation. However, the VTAM topology agent does respond to any DELETE request it receives. (See error discussion, "Responding to CMIP requests" on page 161.)

Other operations

Examples of other OSI operations include create, linked-reply, and other types of ACTION. Although the create and linked-reply operations are valid, there is no situation in which the VTAM topology agent can receive these operations. Other types of ACTIONs can be received by the VTAM topology agent.

The create request is used to create a new instance of a specified object class. For CMIP services to route a create request, an application program must have registered as a **create-handler** for the requested object class. The VTAM topology agent does not register as a create-handler for any object class, so CMIP services never routes a create request to the topology agent.

The linked-reply operation is a request, but it is better described as part of a multiple-message reply. The responses for some requests might require multiple messages. All of the messages except the last message of a multiple message reply are linked-reply operations. A linked-reply message must refer to the original request message for which this message is part of the reply. The VTAM topology agent does not send any requests for which a linked-reply is returned, so the topology agent never receives linked-reply operations. The VTAM topology agent does, however, send linked-reply messages.

As mentioned before, there are a number of action types for the ACTION operation other than *snapshot*. The VTAM topology agent does not support the other action types, but the topology agent responds to any confirmed ACTION request it receives. For a discussion of errors, refer to "Responding to CMIP requests."

Responding to CMIP requests

This section provides an overview of the types of responses, both positive and negative, that the VTAM topology agent provides, given an input CMIP request. Subsequent sections describe the details of monitoring VTAM resources by using the CMIP requests.

A CMIP request message is really a form of a **protocol data unit (PDU)**, as are the various kinds of response messages. The following list provides a summary of the types of PDUs used by the VTAM topology agent:

ROIVapdu
: The **ROIV** message represents a request message and is usually an unsolicited message. In one case, an ROIV represents one of a set of linked-reply messages, but even in this case the ROIV is treated as a request message. The ROIV request messages are either confirmed (requiring a response) or unconfirmed (allowing no response), depending on the particular operation being requested. The linked-reply ROIV message might contain the requested response data, or it might contain an indication that an error has occurred. All requests that are sent to the VTAM topology agent are ROIV messages.

RORSapdu
: The **RORS** message represents a final response message. It is sent only in response to a previous ROIV request message and only if the ROIV request requires a response message. An ROIV request message can have a maximum of one RORS message sent in response. Therefore, if a request requires multiple reply messages, all but the final reply messages must be in the form of ROIV linked-reply messages. The VTAM topology agent sends RORS messages in response to all confirmed requests that it receives, when the subsequent processing is successful. An RORS message is also sent if an error occurred and the error indication was sent as part of a linked-reply ROIV message.

ROERapdu
: The **ROER** message represents a negative response message. It is used to indicate the unsuccessful processing for a request message. For the ROER message to be used for a response, it must be the only message in the response. Therefore, if one or more linked-reply ROIV messages are sent in a response and then an error occurs, the ROER message cannot be used to indicate the processing error. Instead, the error is indicated in an additional linked-reply ROIV, followed by an RORS.

Responding to GET ROIV messages

The VTAM topology agent can receive a GET ROIV request for any VTAM resource, regardless of whether the resource exists. Whether the resource is valid or not, all GET requests are valid, and the agent always responds with either a single positive response (RORS) or a single negative response (ROER).

Responding to CANCEL-GET messages

The CANCEL-GET ROIV request, by virtue of being sent to the VTAM topology agent from CMIP services, *must* refer to an existing, valid GET request. Two responses are generated, an ROER for the GET request that is referred to, indicating that the operation was cancelled, and an RORS for the CANCEL-GET request indicating that this operation was completed successfully. If the CANCEL-GET request is issued after the VTAM topology agent has processed the GET request, then the response to the CANCEL-GET request is an ROER, indicating that this operation could not be processed.

Responding to ACTION ROIV messages

An ACTION ROIV request can be valid or not valid. The VTAM topology agent responds to ACTION requests that are not valid with a single negative response (ROER). Valid ACTION requests are processed, but errors can still occur during that processing. The VTAM topology agent can respond to valid ACTION requests with either positive or negative responses. These responses, however, may not be simple single messages; instead, they may involve a series of messages.

A typical positive response to an ACTION is a number of linked-reply ROIV messages followed by a single ACTION response message (RORS). A negative response to an ACTION request can be more complicated; the negative response may take the form of the ROER, or it can be a linked-reply ROIV message that contains `specificErrorInfo` data, followed by an RORS message.

The determination of which type of error response is used is dependent upon whether any linked-reply ROIV messages with data have been sent. If no linked-reply messages with data have been sent in response, an ROER is used for the error response. If one or more linked-reply messages have been sent in response to this ACTION request, then a linked-reply ROIV containing the `specificErrorInfo` construct is sent, followed by an RORS response message.

EVENT-REPORT, SET, and DELETE messages

EVENT-REPORT messages are always sent as unconfirmed ROIV messages; these ROIVs do not represent linked-replies, and it is not possible to include any error information in them.

A SET ROIV request sent to the VTAM topology agent will result in either an ROER or an RORS message, depending on the specific data in the request.

A DELETE ROIV request sent to the VTAM topology agent will always result in an ROER response message.

Monitoring resources with the ACTION(snapshot) operation

The ACTION(*snapshot*) operation can be used to collect current resource information and to monitor future resource change information. This section describes the general use of ACTION(*snapshot*) for resource monitoring. Chapter 15, "VTAM topology monitoring," on page 171 provides the details of using ACTION(*snapshot*) to monitor specific kinds of VTAM resource data.

ACTION(snapshot) request

Similar to other operations, the ACTION(*snapshot*) request is sent to an object instance. It differs from other operations in that the number of object classes that support the ACTION(*snapshot*) operation is small. The classes that do support ACTION(*snapshot*) represent collections of objects.

The following object classes support the ACTION(*snapshot*):

Object class
Description

snaLocalTopo
Represents the graph object that contains all of the resources owned by a local VTAM. An ACTION request to this object is asking for the following data:
- Local VTAM data
- Lines
- PUs
- Link stations
- Owned NCP data
- Contacted adjacent node data
- APPN and subarea TGs

snaNetwork
Represents the graph object that contains all of the network information known at a VTAM node.

luCollection
Represents the collection object that contains all of the LU information associated with a specific PU or the VTAM host.

logicalUnitIndex
Represents the collection object that contains the instances of a given LU name known at a VTAM node or throughout the network. Note that an ACTION on the *logicalUnitIndex* object is not considered a monitoring function. It more closely resembles the function of the GET operation and is discussed in detail with the GET operation instead of with the various ACTION monitors.

In addition to the object class and object instance, the ACTION request includes a segment called the `actionInfoArg`. This segment is a sequence of three possible fields, of which any specific ACTION request can include a maximum of two of the following fields:

start
Indicates that the ACTION request is to start a new *snapshot* operation. If this field is present in the request, an additional token of information is included in this field that is one of the following:

oneTimeOnly
Indicates that the *snapshot* is requesting only initial data, that is, the monitoring of future changes to resource data is *not* being requested.

ongoing
Indicates that both initial data and update data are being requested; this includes the current resource information and future changes to this data.

stop
Indicates that the ACTION request is to stop a previous start ACTION request. The start and stop fields are mutually exclusive; however, one of these two fields must appear in a valid ACTION(*snapshot*) request. If the stop field is present, an additional mandatory token is included after the stop token that provides the invoke identifier of the start request that is to be terminated.

additional info (addlInfo)
Optionally specifies data that is specific to the target object. For example, both *snapshots* for *snaLocalTopo* and *snaNetwork* might contain the *appnPlusSubareaParm* parameter, which has an information value representing either `appnOnly` or `appnPlusSubarea`. This field applies only if the start field is included.

ACTION(snapshot) response

ACTION(*snapshot*) responses can be any of the following variations on PDU content:

- Single RORS

  If there is no *snapshot* data for the VTAM topology agent to supply and the request is `oneTimeOnly`, the empty set might be returned in RORS message.
- Linked-reply and RORS

  If the VTAM topology agent has *snapshot* data to provide, the response consists of one or more linked-reply ROIV messages followed by an RORS message.
- Single ROER

  If an error occurs and no data linked-reply ROIV messages are sent in response, a single ROER message is sent in response, indicating the failure to process successfully.
- Linked-reply error and RORS

  If one or more linked-reply ROIV messages with data is sent in response before an error occurs, an ROER message cannot be used. In this case, an additional linked-reply ROIV is sent containing an indication of the error, followed by an RORS message.

In a *snapshot* response message containing valid data, the basic unit of information is a sequence of:
- vertex 1 (v1)
- vertex 2 (v2)
- endpoint 1 (e1)

Multiple sets of this sequence (v1, v2, e1) can occur within a *snapshot* response. Each of the three components (v1, v2, e1) has the same basic syntactic structure. However, the semantics and the actual structure of v1, v2, and e1 vary with the different target objects of the *snapshot*.

The structure of each v1, v2, or e1 is as follows:

object Provides the distinguished name (DN) of the primary object instance being

reported in this component. If this component is vertex 1 or endpoint 1, additional object instances might be reported elsewhere in the v1 or e1 string.

class Provides the object class of the instance identified in the object field.

states Provides a condensed, encoded form of the state attributes for the object instance identified in the object field. The attributes are given in the OCTET string form instead of the full attribute form. The state attributes, their positions in the OCTET string, and the possible values are:

Octet	State
0	operationalState
	00 disabled
	01 enabled
	FF N/A or unchanged
1	usageState
	00 idle
	01 active
	02 busy
	FF N/A or unchanged
2	administrativeState
	00 locked
	01 unlocked
	02 shuttingDown
	FF N/A or unchanged
3	availabilityStatus
	Note that this attribute represents a SET, so it might have multiple values; each value shown below represents a bit position. To show multiple values, the bits representing the values are logically ORed together.
	00 noStatus
	01 notInstalled
	02 degraded
	04 dependency
	08 offDuty
	10 offLine
	20 powerOff
	40 failed
	80 inTest
	FF N/A or unchanged
4	proceduralStatus
	00 no status
	08 terminating
	10 reporting
	20 initializing
	40 not initialized
	80 initialization required
	FF N/A or unchanged
5	unknownStatus
	00 false
	01 true
	FF N/A or unchanged

6 nativeStatus

00	Active
01	Active with sessions
02	Inactive
03	Never active
04	pending active
05	pending inactive
06	Connectable
07	Routable
09	Congested
0A	Released
0B	Reset
0C	Inop
FF	N/A or unchanged

info Provides an optional set of attributes associated with the object instance identified in the object field. The list of attributes provided varies with the target *snapshot* object.

moreInfo
Provides for any additional information that is necessary; for example, for the vertex 1 of a *snaLocalTopo snapshot* this field might contain a set of object data specifying a *port* object.

reason Indicates why the *snapshot* update is being sent:

Value	Description
deleted	Object is deleted.
addOrUpdate	Object is added or changed. The default is addOrUpdate.

ACTION(snapshot) initial data

The response data that is common to both an `ongoing` and a `oneTimeOnly` *snapshot* is called *initial data*. The initial data provides the immediate view or snapshot of the current resource data. For a `oneTimeOnly` *snapshot*, the initial data is the entire set of data returned to the manager application program. For an `ongoing` *snapshot*, the initial data is returned first, followed later by *update data*.

Initial data is returned in linked-reply ROIV messages, with the number of messages varying according to *snapshot* type and configuration. The minimum number of linked-replies is one; there is no maximum number.

When all initial data linked-replies have been sent, the VTAM topology agent must notify the manager application program that the initial data is complete. The VTAM topology agent provides two ways for a manager application program to determine that initial data is complete:

- If the *snapshot* is `ongoing`, after all linked-replies with initial data are sent, the VTAM topology agent sends one additional linked-reply message with no *snapshot* data in it, called the empty set linked-reply. The purpose of this message is to indicate that the transfer of initial data is now complete. This special linked-reply is identified by the value of the actionReplyInfo field being `()`. All (v1,v2,e1) data is reported in the actionReplyInfo section of the *snapshot* response, so the absence of data in this field indicates no more initial data.

- If the *snapshot* is `oneTimeOnly`, following the linked-replies of initial data, an empty set linked-reply is sent to indicate initial data transfer is complete. Next, an RORS message is sent to indicate that processing for the entire request is complete.

ACTION(snapshot) update data

Snapshot updates consist of:

- Changes to resource information that was previously reported in initial data
- Resource information not reported in initial data that is newly defined or learned by VTAM

Update data is reported only for `ongoing` *snapshots*. It is never reported for `oneTimeOnly` *snapshots*. Other than being reported later than initial data, following the empty set linked-reply, there is little difference between initial data and update data. The syntax is the same, but the VTAM topology agent generally provides less data in the update data than in the initial data.

The update data is reported in linked-replies that might contain multiple sets of (v1, v2, e1) data. When a topology update occurs, an update is generated and formatted into a response string. However, the VTAM topology agent does not immediately send every update string to the manager application program. Instead, the VTAM topology agent attempts to use storage efficiently by filling the *snapshot* buffer. The VTAM topology agent might wait a short period of time to see if more update data is generated. If more updates are generated, the updates are added to the existing *snapshot* response. If no updates occur within the time interval (1 second), the existing *snapshot* response is sent to the manager application program.

Update data is generated by the VTAM topology agent for these reasons:

- An object is created.

  A resource that is within the scope of a given *snapshot* becomes known to VTAM and there is an `ongoing` *snapshot* in the update phase. The creation of the object is reported as update data.
- An object is deleted.

  A resource previously reported within a given *snapshot* is no longer known to VTAM. The resource might be deleted because a major node was inactivated or a connection was removed. This change is reported as an object deletion in update data.
- An object changes state.

  A resource previously reported within a given *snapshot* has changed state. It is important to note that the state that has changed is the VTAM internal state; this state is mapped to the set of state attributes that are reported for the object. Since several VTAM internal states are mapped to the same set of state attribute values, there is no guarantee that any of the seven state attributes changed, although it is likely.

Although VTAM in general does not report updates when attribute values change, there are exceptions. In cases where significant data associated with a reported resource has changed, updates are sent to report new attribute values, even if the state of the resource does not change.

ACTION(snapshot) update merging

As noted previously, one of the reasons for reporting topology update data is resource state changes. Of the state changes that occur for resources, many of the resulting states are transient in nature. That is, the resource is in transition from one state to another, but the transition is through a series of intermediate states. These intermediate states are usually brief and are considered less important than the resting states.

Since the number of updates reporting transient state changes can be large, the VTAM topology agent suppresses the intermediate updates. This process is called *merging*, since the intermediate updates are merged instead of discarded.

Updates that can be merged are:
- *snaLocalTopo*
- *snaNetwork*, for CDRMs only
- *luCollection* (with some exceptions)

luCollection updates for independent LUs in a *snapshot* directed at a specific PU are not merged. *snaNetwork* updates for APPN network data are not merged.

Updates for resources moving to transient states are merged until an update is received that shows the resource moving to a resting state. The resting state update is merged, and the final, merged update is sent.

It is possible for resources to stay in transient states too long; for example, when an error occurs and a resource is hung in a state that is not a resting state. The VTAM topology agent periodically looks for updates that have been held too long; when these are found, they are sent.

The UPDDELAY start option controls the maximum length of time that VTAM waits before looking for resource updates that have been in transient states too long. This start option specifies the maximum number of seconds to wait before looking for resources that are hung. Note that decreasing the value for UPDDELAY might force VTAM to look for these resources more often, but does not necessarily imply that resources are reported any faster. The time specified in the UPDDELAY start option does not affect the computing of whether a resource has stayed in a transient state too long. That computing is not controllable and is based primarily on recent updates statistics.

Although a given type of *snapshot* update is eligible to be merged, in some cases, individual updates are not merged.

EVENT-REPORT data is also subject to the merge process.

By design, the merge process suppresses intermediate state data; however, there is one case where the loss of intermediate state data is not acceptable. The *availabilityStatus* attribute reports the value of `dependency` when a higher-level resource is in transition. When updates are merged, the newest state data generally replaces the older state data, but for the dependency information this is not the case. The VTAM topology agent preserves the dependency information from all updates that are merged to a single update by performing the logical OR operation on the dependency information from all updates. The result is that if any of the set of merged updates have the dependency flag set for a resource, the reported (merged) update reports the dependency in the *availabilityStatus* attribute.

ACTION(snapshot) termination

If the start request for the ACTION(*snapshot*) is for a `oneTimeOnly` *snapshot*, then it is not necessary to issue a stop request. The `oneTimeOnly` *snapshot* stops itself when the initial data transfer is complete. The `ongoing` *snapshot*s, however, must be explicitly stopped by using an additional ACTION(*snapshot*) request that specifies stop instead of start. The stop request must indicate which *snapshot* request should be terminated by including the invoke identifier of the start *snapshot* request.

Upon receiving a stop ACTION(*snapshot*) request, the VTAM topology agent suspends the reporting of update data, responds to both the start request and the stop request, and terminates processing for both requests. Note that the stop request is not processed until the transfer of the initial data for the target *snapshot* is complete.

Chapter 15. VTAM topology monitoring

This chapter describes the details of the specific monitoring capabilities of the VTAM topology agent. The following topics are included:
- Requesting and monitoring network data (snaNetwork)
- Requesting and monitoring local topology (snaLocalTopo)
- Requesting and monitoring LU data (luCollection)
- Monitoring resources through event reports

Requesting and monitoring network data (snaNetwork)

This section contains the following topics:

Overview

This section describes the action that is used to request monitoring and stop monitoring network data for the *snaNetwork* managed object class.

Management of the network requires that a manager application program be able to request the names of all nodes and APPN transmission groups and virtual routes that connect any two nodes and to be able to monitor their status.

Operations against *snaNetwork* can be directed at the following VTAM topology agent host node types:
- Interchange node
- Network node
- Migration data host
- Pure type 5 node

Note: Pure end nodes cannot provide *snaNetwork* data and fail the request with an ROER response.

Action request

A *snapshot* action request is used to request network data from the VTAM topology agent. The action is sent as an m-Action-Confirmed operation.

The manager application program can request that any future updates to the *snaNetwork* *snapshot* object to be returned, as they occur by specifying `ongoing` in the request. The network data can be requested without updates by specifying the `oneTimeOnly` value in the request.

The manager application program can specify the *appnPlusSubareaParm* in the additional information field. The value, either 0 or 1, represents either `appnOnly` (0),

which means to request APPN network data, or `appnPlusSubarea` (1), which means to request both APPN and subarea network data. The default value is `appnOnly` if the *appnPlusSubareaParm* is not specified.

The target resource is the only object instance of the *snaNetwork* object class at the VTAM topology agent. Following is the example of an `ongoing`, `appnPlusSubarea` *snaNetwork* *snapshot* request:

```
                msg CMIP-1.ROIVapdu (invokeID 196610, opera
tion-value 7, argument (baseManagedObjectClass 1.3.18.0.0.21
51,baseManagedObjectInstance (distinguishedName (RelativeDis
tinguishedName (AttributeValueAssertion (attributeType 1.3.1
8.0.2.4.6, atributeValue NETA)), RelativeDistinguishedName (
AttributeValueAssertion (attributeType 1.3.18.0.0.2032, attr
ibuteValue SSCP1A)), RelativeDistinguishedName (AttributeVal
ueAssertion (attributeType 1.3.18.0.0.2216, attributeValue (
string "SnaNetwork"))))),actionInfo (actionType 1.3.18.0.0.2
222,actionInfoArg (start ongoing, addlInfo ((identifier 1.3.
18.0.0.2164, significance TRUE, information 1))))))
```

Initial data response

The interchange node or network node provides the APPN network data that is current at the time the operation is processed. The APPN network data includes information about network nodes, interchange nodes, border nodes, virtual routing nodes, and transmission groups (TGs) that connect the APPN nodes. A connection between nodes A and B is reported once for each direction: from Node A to Node B and from Node B to Node A.

The migration data host nodes, type 5 nodes and interchange nodes provide the subarea network data that is current at the time the operation is processed. The subarea network data includes information about the local VTAM agent host, cross-domain resource managers, and virtual routes that connect the subarea nodes.

Interchange nodes are the only nodes that provide both APPN and subarea data.

If the `oneTimeOnly` *snapshot* action is requested, the initial data is returned in action linked-replies. To indicate that the initial data for the entire set of network data has been returned, the VTAM topology agent sends an additional ROIV action linked-reply that is an empty set linked-reply. Following the empty set is an RORS message.

If the `ongoing` *snapshot* action is requested, all the initial data is returned in action linked-replies, just as for the `oneTimeOnly` *snapshot* action and is followed by an empty-set linked-reply. The VTAM topology agent is then ready to process updates for the *snaNetwork* object.

Update data response

When the `ongoing` *snapshot* action has been issued and is currently in effect, the following changes cause updates to the *snaNetwork* object, which result in the sending of a *snaNetwork* *snapshot* linked-reply:

- Any status changes for a node or APPN transmission group (TG)
- Changes in the characteristics of a node or APPN TG
- Creation, deletion, or state change of a cross-domain resource manager

Updates for the *snaNetwork* object for subarea network data might be merged with related updates by the VTAM topology agent before being written to the *snapshot* linked-replies.

Action termination

The VTAM topology agent terminates an ongoing *snaNetwork* *snapshot* action under the following conditions :

- A stop *snapshot* action request is received.
- An error occurs during *snapshot* processing in VTAM.
- The association between the local CMIP services and the manager application program's CMIP services terminates.

The following is an example of a valid stop *snapshot* action request :

```
            msg CMIP-1.ROIVapdu (invokeID 196612,
 operation-value 7, argument (baseManagedObjectCl
ass 1.3.18.0.0.2151, baseManagedObjectInstance (d
istinguishedName (RelativeDistinguishedName (Attr
ibuteValueAssertion (attributeType 1.3.18.0.2.4.6
, attributeValue "NETA")), RelativeDistinguishedN
ame (AttributeValueAssertion (attributeType 1.3.1
8.0.0.2032, attributeValue "SSCP1A")), RelativeDi
stinguishedName (AttributeValueAssertion (attribu
teType 1.3.18.0.0.2216, attributeValue (string "S
naNetwork"))))), actionInfo (actionType 1.3.18.0.
0.2222, actionInfoArg (stop 196610))))
```

The stop request in the previous example looks much like the associated start request except for the `actionInfoArg` portion of the request. For the stop request the `stop` keyword is used along with the invoke identifier of the start request that is to be terminated.

The result of the VTAM topology agent's processing of a stop request is three messages :

- An empty-set linked-reply for the start request
- An RORS response to the start request
- An RORS response to the stop request

The following shows examples of these three messages:

First, the empty-set linked reply:

(Note that the associated invoke identifier in a linked-reply is given in the `linked-ID` field.)

```
                    msg CMIP-1.ROIVapdu (invokeID
 1, linked-ID 196610, operation-value 2, argument
 (actionResult (managedObjectClass 1.3.18.0.0.215
1, managedObjectInstance (distinguishedName (Rela
tiveDistinguishedName (AttributeValueAssertion (a
ttributeType 1.3.18.0.2.4.6, attributeValue "NETA
")), RelativeDistinguishedName (AttributeValueAss
ertion (attributeType 1.3.18.0.0.2032, attributeV
alue "SSCP1A")), RelativeDistinguishedName (Attri
buteValueAssertion (attributeType 1.3.18.0.0.2216
, attributeValue (string "SnaNetwork"))))), actio
nReply (actionType 1.3.18.0.0.2222, actionReplyIn
fo ())))
```

Next, the RORS for the start request :

```
                    msg CMIP-1.RORSapdu (invokeID
 196610)
```

Finally, the RORS for the stop request:

```
                    msg CMIP-1.RORSapdu (invokeID
 196612, resultOption (operation-value 7, result
(managedObjectClass 1.3.18.0.0.2151, managedObjec
tInstance (distinguishedName (RelativeDistinguish
edName (AttributeValueAssertion (attributeType 1.
3.18.0.2.4.6, attributeValue "NETA")), RelativeDi
stinguishedName (AttributeValueAssertion (attribu
teType 1.3.18.0.0.2032, attributeValue "SSCP1A"))
, RelativeDistinguishedName (AttributeValueAssert
ion (attributeType 1.3.18.0.0.2216, attributeValu
e (string "SnaNetwork"))))), actionReply (actionT
ype 1.3.18.0.0.2222, actionReplyInfo ())))
```

snaNetwork snapshot data (APPN data)

For APPN network topology data, the linked-replies for *snaNetwork* are made up of multiple instances of the following sequence:

```
vertex1    --Origin node of the transmission group
vertex2    --Destination node of the transmission group
endpoint1  --Transmission group between vertex1 and vertex2
```

In general, the data structure of vertex1, vertex2, and endpoint1 is as follows:

```
vertex1
        object --object distinguished name
        class  --object class
        states --OSI states of this object
        info
           resourceSequenceNumber       --object attribute
           appnNodeCapabilities         --object attribute
           extendedAppnNodeCapabilities --object attribute
                                             --(reported only for central
                                             --directory server nodes)

vertex2
        object --object distinguished name
        class  --object class

endpoint1
        object --object distinguished name
        class  --object class (APPN TG)
        states --OSI states of this object
        info,
            resourceSequenceNumber  --object attribute
            appnTGCapabilities      --object attribute
            cp-cpSessionSupport     --object attribute
```

The format of vertex1 differs according to the data received. When vertex2 or endpoint1 is the main reason for an update, vertex1 shows only the following:

```
vertex1
        object  --object distinguished name
        class   --object class
```

The following list includes descriptions of what is contained in vertex1, vertex2, and endpoint1.

vertex1

Data reported for a single node object for either initial data or for a single update for the object. Vertex 1 is considered the origin of the TG specified in endpoint 1.

object Distinguished name of origin node resource

class Objects related to the monitored node are reported under the following object classes:
- *appnNN*
- *interchangeNode*
- *virtualRoutingNode*

states 14-character string for the OSI states:
- *operationalState*
- *usageState*
- *administrativeState*
- *availabilityStatus*
- *proceduralStatus*
- *unknownStatus*
- *nativeStatus* (*nativeStatus* is always N/A for the APPN *snaNetwork* data.)

info Set of attributes for the node object. This field is missing from vertex1 if vertex2 or endpoint1 is the main reason for the update.

vertex2

Contains all data reported for a single node object for either initial data or a single update for the object. Vertex 2 is considered the destination of the TG specified in endpoint 1.

object Distinguished name of destination node resource.

class Objects related to the monitored node are reported under the following object classes:
- *appnNN*
- *virtualRoutingNode*

Note that interchange nodes are reported in class *appnNN* in vertex 2.

endpoint1

Contains data for a TG that connects vertex1 and vertex2.

object Distinguished name of TG resource.

class Monitored transmission group object is reported under the *appnTransmissionGroup* object class.

states Consists of a 14-character string for the OSI states. (*nativeStatus* is always N/A for the APPN *snaNetwork* data.)

info Set of attributes for the transmission group object.

Note that the `reason` field is always omitted for APPN data and should always assume the default value of `addOrUpdate`.

snaNetwork snapshot data (subarea data)

For subarea network topology data, the linked-replies for *snaNetwork* contain data made up of multiple instances of the following sequence:

```
vertex1    --Adjacent SSCP (CDRM)
vertex2    --Local VTAM
endpoint1  --Virtual route supporting active CDRM
```

VTAM provides the long form of vertex1 when it reports initial data or object creation such as for the activation of a new CDRM major node. VTAM provides the short form of vertex1 when it reports changes or deletions.

In general, the data structure of vertex1, vertex2, and endpoint1 is as follows:

```
vertex1
        object  --object distinguished name
        class   --object class (CDRM)
        states  --OSI states of this object
        info
           dependencies  --object attribute
           realSSCPname  --object attribute
        reason  --reason for this vertex1 to be reported

vertex2
        object  --object distinguished name (local VTAM)
        class   --object class

endpoint1
        object  --object distinguished name
        class   --object class (VR)
```

The following list explains what the fields contain.

vertex1
Contains all data reported for a single node object for either initial data or a single update for the object (CDRM).

object Distinguished name of CDRM resource

class Monitored node objects are reported under the *crossDomainResourceManager* object class.

states Consists of a 14-character string for the OSI states

info Set of attributes for the CDRM related object.

reason Indicates why the *snapshot* update is being sent.

Note: The `reason` field is omitted when the intended value is `addOrUpdate`.

Value	Description
deleted	Object is deleted.
addOrUpdate	Object is added or changed. The default is addOrUpdate.

vertex2
Reports on the local VTAM agent host.

object Distinguished name of local VTAM agent host.

class Always reported as *t5node* object class.

endpoint1
Contains all data reported for a single virtual route object for either initial data or a single update for the virtual route.

object Distinguished name of the virtual route.

class Monitored virtual route objects are reported under the *virtualRoute* object class.

To see the vertex1 entries that are included in each type of subarea *snapshot* action, refer to Table 12 on page 177.

Table 12. vertex1 entries for CDRM reported objects. Object creation means the creation of a new CDRM.

	Initial data	Object creation	Object state change	Object attribute change	Object deletion
object	X	X	X	X	X
class	X	X	X	X	X
states	X	N/A	X	X	N/A
info	X	X	X	X	X
dependencies	X	X	X	X	N/A
realSSCPname	X	X	X	N/A	N/A
reason	N/A	N/A	N/A	N/A	X

snaNetwork snapshot example

The following example shows the initial data response for `appnPlusSubarea` *snaNetwork snapshot* of the following configuration, where SSCP2A is defined to SSCP1A as a CDRM and an APPN TG is active between SSCP1A and SSCP2A.

```
(Interchange node)
       SSCP1A -------------------- SSCP2A
                 APPN TG 21     (Interchange node)
```

```
                  msg CMIP-1.ROIVapdu (invokeID
 131074, linked-ID 196610, operation-value 2, arg
ument (actionResult (managedObjectClass 1.3.18.0.
0.2151, managedObjectInstance (distinguishedName
(RelativeDistinguishedName (AttributeValueAsserti
on (attributeType 1.3.18.0.2.4.6, attributeValue
"NETA")), RelativeDistinguishedName (AttributeVal
ueAssertion (attributeType 1.3.18.0.0.2032, attri
buteValue "SSCP1A")), RelativeDistinguishedName (
AttributeValueAssertion (attributeType 1.3.18.0.0
.2216, attributeValue (string "SnaNetwork"))))),
actionReply (actionType 1.3.18.0.0.2222, actionRe
plyInfo ((vertex1 (object (distinguishedName (Rel
ativeDistinguishedName (AttributeValueAssertion (
attributeType 1.3.18.0.2.4.6, attributeValue "NET
A")), RelativeDistinguishedName (AttributeValueAs
sertion (attributeType 1.3.18.0.0.2032, attribute
Value "SSCP1A")))), class 1.3.18.0.0.1826, states
 010101000000FF, info (Attribute (attributeId 1.3
.18.0.0.2019, attributeValue 2), Attribute (attri
buteId 1.3.18.0.0.1940, attributeValue 3348))), v
ertex2 (object (distinguishedName (RelativeDistin
guishedName (AttributeValueAssertion (attributeTy
pe 1.3.18.0.2.4.6, attributeValue "NETA")), Relat
iveDistinguishedName (AttributeValueAssertion (at
tributeType 1.3.18.0.0.2032, attributeValue "SSCP
2A")))), class 1.3.18.0.0.1822), endpoint1 (objec
t (distinguishedName (RelativeDistinguishedName (
AttributeValueAssertion (attributeType 1.3.18.0.2
.4.6, attributeValue "NETA")), RelativeDistinguis
hedName (AttributeValueAssertion (attributeType 1
.3.18.0.0.2032, attributeValue "SSCP1A")), Relati
veDistinguishedName (AttributeValueAssertion (att
ributeType 1.3.18.0.0.2044, attributeValue "21.NE
TA.SSCP2A")))), class 1.3.18.0.0.1823, states 010
101000000FF, info (Attribute (attributeId 1.3.18.
0.0.2019, attributeValue 2), Attribute (attribute
```

```
Id 1.3.18.0.0.1941, attributeValue 00), Attribute
 (attributeId 1.3.18.0.0.1958, attributeValue TRU
E)))), (vertex1 (object (distinguishedName (Relat
iveDistinguishedName (AttributeValueAssertion (at
tributeType 1.3.18.0.2.4.6, attributeValue "NETA"
)), RelativeDistinguishedName (AttributeValueAsse
rtion (attributeType 1.3.18.0.0.2032, attributeVa
lue "SSCP2A")))), class 1.3.18.0.0.1826, states 0
10101000000FF, info (Attribute (attributeId 1.3.1
8.0.0.2019, attributeValue 2), Attribute (attribu
teId 1.3.18.0.0.1940, attributeValue 3348))), ver
tex2 (object (distinguishedName (RelativeDistingu
ishedName (AttributeValueAssertion (attributeType
 1.3.18.0.2.4.6, attributeValue "NETA")), Relativ
eDistinguishedName (AttributeValueAssertion (attr
ibuteType 1.3.18.0.0.2032, attributeValue "SSCP1A
")))), class 1.3.18.0.0.1822), endpoint1 (object
(distinguishedName (RelativeDistinguishedName (At
tributeValueAssertion (attributeType 1.3.18.0.2.4
.6, attributeValue "NETA")), RelativeDistinguishe
dName (AttributeValueAssertion (attributeType 1.3
.18.0.0.2032, attributeValue "SSCP2A")), Relative
DistinguishedName (AttributeValueAssertion (attri
buteType 1.3.18.0.0.2044, attributeValue "21.NETA
.SSCP1A")))), class 1.3.18.0.0.1823, states 01010
1000000FF, info (Attribute (attributeId 1.3.18.0.
0.2019, attributeValue 2), Attribute (attributeId
 1.3.18.0.0.1941, attributeValue 00), Attribute (
attributeId 1.3.18.0.0.1958, attributeValue TRUE)
))), (vertex1 (object (distinguishedName (Relativ
eDistinguishedName (AttributeValueAssertion (attr
ibuteType 1.3.18.0.2.4.6, attributeValue "NETA"))
, RelativeDistinguishedName (AttributeValueAssert
ion (attributeType 1.3.18.0.0.2032, attributeValu
e "SSCP1A")), RelativeDistinguishedName (Attribut
eValueAssertion (attributeType 1.3.18.0.0.2032, a
ttributeValue "NETA.SSCP1A")))), class 1.3.18.0.0
.2278, states 01010100000000, info (Attribute (at
tributeId 1.3.18.0.0.2194, attributeValue (depend
ents (and (Dependents (item (distinguishedName (R
elativeDistinguishedName (AttributeValueAssertion
 (attributeType 1.3.18.0.2.4.6, attributeValue "N
ETA")), RelativeDistinguishedName (AttributeValue
Assertion (attributeType 1.3.18.0.0.2032, attribu
teValue "SSCP1A")), RelativeDistinguishedName (At
tributeValueAssertion (attributeType 1.3.18.0.0.2
272, attributeValue "CDRM.CDRM1A"))))), Dependent
s (item (distinguishedName (RelativeDistinguished
Name (AttributeValueAssertion (attributeType 1.3.
18.0.2.4.6, attributeValue "NETA")), RelativeDist
inguishedName (AttributeValueAssertion (attribute
Type 1.3.18.0.0.2032, attributeValue "SSCP1A"))))
))))), Attribute (attributeId 1.3.18.0.0.5246, at
tributeValue "NETA.SSCP1A"))), vertex2 (object (d
istinguishedName (RelativeDistinguishedName (Attr
ibuteValueAssertion (attributeType 1.3.18.0.2.4.6
, attributeValue "NETA")), RelativeDistinguishedN
ame (AttributeValueAssertion (attributeType 1.3.1
8.0.0.2032, attributeValue "SSCP1A")))), class 1.
3.18.0.0.1845)), (vertex1 (object (distinguishedN
ame (RelativeDistinguishedName (AttributeValueAss
ertion (attributeType 1.3.18.0.2.4.6, attributeVa
lue "NETA")), RelativeDistinguishedName (Attribut
eValueAssertion (attributeType 1.3.18.0.0.2032, a
ttributeValue "SSCP1A")), RelativeDistinguishedNa
me (AttributeValueAssertion (attributeType 1.3.18
.0.0.2032, attributeValue "NETA.SSCP2A")))), clas
```

```
s 1.3.18.0.0.2278, states 00000100400003, info (A
ttribute (attributeId 1.3.18.0.0.2194, attributeV
alue (dependents (and (Dependents (item (distingu
ishedName (RelativeDistinguishedName (AttributeVa
lueAssertion (attributeType 1.3.18.0.2.4.6, attri
buteValue "NETA")), RelativeDistinguishedName (At
tributeValueAssertion (attributeType 1.3.18.0.0.2
032, attributeValue "SSCP1A")), RelativeDistingui
shedName (AttributeValueAssertion (attributeType
1.3.18.0.0.2272, attributeValue "CDRM.CDRM1A"))))
), Dependents (item (distinguishedName (RelativeD
istinguishedName (AttributeValueAssertion (attrib
uteType 1.3.18.0.2.4.6, attributeValue "NETA")),
RelativeDistinguishedName (AttributeValueAssertio
n (attributeType 1.3.18.0.0.2032, attributeValue
"SSCP1A"))))))))), Attribute (attributeId 1.3.18.
0.0.5246, attributeValue ""))), vertex2 (object (
distinguishedName (RelativeDistinguishedName (Att
ributeValueAssertion (attributeType 1.3.18.0.2.4.
6, attributeValue "NETA")), RelativeDistinguished
Name (AttributeValueAssertion (attributeType 1.3.
18.0.0.2032, attributeValue "SSCP1A")))), class 1
.3.18.0.0.1845)))))))
```

The linked-reply in the example, identified by the operation value being 2, contains a set of 4 instances of the (v1,v2,e1) sequence, although not all instances of the sequence contain all fields of the sequence. The following is a summary of the contents of the 4 sequences:

First sequence: (v1,v2,e1) APPN data

```
vertex 1 :   NETA;SSCP1A

               Class   : interchangeNode
               States  :
                  Operational State    : Enabled
                  Usage State          : Active
                  Administrative State : Unlocked
                  Availability Status  : No Status
                  Procedural Status    : No Status
                  Unknown Status       : False
                  Native Status        : N/A
               Info :
                  resourceSequenceNumber        : 2
                  appnNodeCapabilities          : 3348

vertex 2 :   NETA;SSCP2A

               Class   : appnNN

endpoint 1 : NETA;SSCP1A;21.NETA.SSCP2A

               Class   : appnTransmissionGroup
               States  :
                   Operational State    : Enabled
                   Usage State          : Active
                   Administrative State : Unlocked
                   Availability Status  : No Status
                   Procedural Status    : No Status
                   Unknown Status       : False
                   Native Status        : N/A
               Info :
                   resourceSequenceNumber        : 2
                   appnTGcapabilities            : 00
                   cp-cpSessionSupport           : TRUE
```

This sequence represents the connection from the local node to partner node. This sequence shows:

- In vertex 1: name, class and attributes of the local node (origin of TG). SSCP1A is not a central directory server, so extendedAppnNodeCapabilities is not reported.
- In vertex 2: name and class of the partner node (destination of TG).
- In endpoint 1: name, class and attributes of the APPN transmission group (TG number : 21)

Second sequence: (v1,v2,e1) APPN data

```
vertex 1 :   NETA;SSCP2A

               Class   : interchangeNode
               States  :
                  Operational State    : Enabled
                  Usage State          : Active
                  Administrative State : Unlocked
                  Availability Status  : No Status
                  Procedural Status    : No Status
                  Unknown Status       : False
                  Native Status        : N/A
               Info :
                  resourceSequenceNumber        : 2
                  appnNodeCapabilities          : 3348

vertex 2 :   NETA;SSCP1A

               Class   : appnNN

endpoint 1 : NETA;SSCP2A;21.NETA.SSCP1A

               Class   : appnTransmissionGroup
               States  :
                   Operational State    : Enabled
                   Usage State          : Active
                   Administrative State : Unlocked
                   Availability Status  : No Status
                   Procedural Status    : No Status
                   Unknown Status       : False
                   Native Status        : N/A
               Info :
                   resourceSequenceNumber        : 2
                   appnTGcapabilities            : 00
                   cp-cpSessionSupport           : TRUE
```

This sequence represents the connection from the partner node to local node. This sequence shows:

- In vertex 1: name, class and attributes of the partner node (origin of TG). SSCP2A is not a central directory server so extendedAppnNodeCapabilities is not reported.
- In vertex 2: name and class of the local node (destination of TG).
- In endpoint 1: name, class and attributes of the APPN transmission group (TG number : 21). Note that APPN transmission groups are unidirectional and this is a different TG than reported in the first sequence.

Third sequence: (v1,v2) subarea data

```
vertex1 :    NETA;SSCP1A;NETA.SSCP1A

               Class   : CrossDomainResourceManager
               States :
```

```
                    Operational State    : Enabled
                    Usage State          : Active
                    Administrative State : Unlocked
                    Availability Status  : No Status
                    Procedural Status    : No Status
                    Unknown Status       : False
                    Native Status        : ACTIVE

                Info :
                    realSSCPname         : NETA.SSCP1A
                    dependencies         : NETA;SSCP1A;CDRM.CDRM1A
                                           NETA;SSCP1A

vertex2 :    NETA;SSCP1A

               Class   : t5Node
```

This sequence shows the name and state of local agent host CDRM. This sequence shows:

- The long form of vertex1. The first object in Dependencies is a *definitionGroup* representing the major node where the CDRM is defined.
- The **only** form of vertex2, representing the local VTAM agent host. Note the class is reported as *t5Node* for the subarea topology even though the actual host type is interchange node (as shown in the first sequence).

Fourth sequence: (v1,v2) subarea data

```
vertex1 :    NETA;SSCP1A;NETA.SSCP2A

               Class   : CrossDomainResourceManager
               States :
                    Operational State    : Disabled
                    Usage State          : Idle
                    Administrative State : Unlocked
                    Availability Status  : No Status
                    Procedural Status    : Not Initialized
                    Unknown Status       : False
                    Native Status        : NEVAC

               Info :
                    realSSCPname         : NETA.SSCP1A
                    dependencies         : NETA;SSCP1A;CDRM.CDRM1A
                                           NETA;SSCP1A

vertex2 :    NETA;SSCP1A

               Class   : t5Node
```

This sequence shows the name and state of CDRM for SSCP2A (a cross domain host). This sequence shows:

- The long form of vertex1. SSCP2A is not an active CDRM (no CDRM-CDRM session) so no virtual route is reported.
- The **only** form of vertex2, representing the local VTAM agent host.

The following example shows the APPN network topology update data for the ongoing *snaNetwork* snapshot. The update was caused by inactivating the line and PU connecting the two hosts. In this example only APPN topology changes are reported because this is an APPN connection.

```
                    msg CMIP-1.ROIVapdu (invokeID
  131076, linked-ID 196610, operation-value 2, arg
 ument (actionResult (managedObjectClass 1.3.18.0.
 0.2151, managedObjectInstance (distinguishedName
 (RelativeDistinguishedName (AttributeValueAsserti
```

```
on (attributeType 1.3.18.0.2.4.6, attributeValue
"NETA")), RelativeDistinguishedName (AttributeVal
ueAssertion (attributeType 1.3.18.0.0.2032, attri
buteValue "SSCP1A")), RelativeDistinguishedName (
AttributeValueAssertion (attributeType 1.3.18.0.0
.2216, attributeValue (string "SnaNetwork"))))),
actionReply (actionType 1.3.18.0.0.2222, actionRe
plyInfo ((vertex1 (object (distinguishedName (Rel
ativeDistinguishedName (AttributeValueAssertion (
attributeType 1.3.18.0.2.4.6, attributeValue "NET
A")), RelativeDistinguishedName (AttributeValueAs
sertion (attributeType 1.3.18.0.0.2032, attribute
Value "SSCP1A")))), class 1.3.18.0.0.1822), verte
x2 (object (distinguishedName (RelativeDistinguis
hedName (AttributeValueAssertion (attributeType 1
.3.18.0.2.4.6, attributeValue "NETA")), RelativeD
istinguishedName (AttributeValueAssertion (attrib
uteType 1.3.18.0.0.2032, attributeValue "SSCP2A")
))), class 1.3.18.0.0.1822), endpoint1 (object (d
istinguishedName (RelativeDistinguishedName (Attr
ibuteValueAssertion (attributeType 1.3.18.0.2.4.6
, attributeValue "NETA")), RelativeDistinguishedN
ame (AttributeValueAssertion (attributeType 1.3.1
8.0.0.2032, attributeValue "SSCP1A")), RelativeDi
stinguishedName (AttributeValueAssertion (attribu
teType 1.3.18.0.0.2044, attributeValue "21.NETA.S
SCP2A")))), class 1.3.18.0.0.1823, states 0101010
00000FF, info (Attribute (attributeId 1.3.18.0.0.
2019, attributeValue 4), Attribute (attributeId 1
.3.18.0.0.1941, attributeValue 00), Attribute (at
tributeId 1.3.18.0.0.1958, attributeValue TRUE)))
), (vertex1 (object (distinguishedName (RelativeD
istinguishedName (AttributeValueAssertion (attrib
uteType 1.3.18.0.2.4.6, attributeValue "NETA")),
RelativeDistinguishedName (AttributeValueAssertio
n (attributeType 1.3.18.0.0.2032, attributeValue
"SSCP1A")))), class 1.3.18.0.0.1822), vertex2 (ob
ject (distinguishedName (RelativeDistinguishedNam
e (AttributeValueAssertion (attributeType 1.3.18.
0.2.4.6, attributeValue "NETA")), RelativeDisting
uishedName (AttributeValueAssertion (attributeTyp
e 1.3.18.0.0.2032, attributeValue "SSCP2A")))), c
lass 1.3.18.0.0.1822), endpoint1 (object (disting
uishedName (RelativeDistinguishedName (AttributeV
alueAssertion (attributeType 1.3.18.0.2.4.6, attr
ibuteValue "NETA")), RelativeDistinguishedName (A
ttributeValueAssertion (attributeType 1.3.18.0.0.
2032, attributeValue "SSCP1A")), RelativeDistingu
ishedName (AttributeValueAssertion (attributeType
 1.3.18.0.0.2044, attributeValue "21.NETA.SSCP2A"
)))), class 1.3.18.0.0.1823, states 000001000000F
F, info (Attribute (attributeId 1.3.18.0.0.2019,
attributeValue 6), Attribute (attributeId 1.3.18.
0.0.1941, attributeValue 00), Attribute (attribut
eId 1.3.18.0.0.1958, attributeValue TRUE)))))))))
```

First sequence: (v1,v2,e1) APPN data

```
vertex 1 :   NETA;SSCP1A

               Class   : appnNN

vertex 2 :   NETA;SSCP2A

               Class   : appnNN

endpoint 1 : NETA;SSCP1A;21.NETA.SSCP2A
```

```
                Class   : appnTransmissionGroup
                States  :
                    Operational State    : Enabled
                    Usage State          : Active
                    Administrative State : Unlocked
                    Availability Status  : No Status
                    Procedural Status    : No Status
                    Unknown Status       : False
                    Native Status        : N/A
                Info :
                    resourceSequenceNumber        : 4
                    appnTGcapabilities            : 00
                    cp-cpSessionSupport           : TRUE
```

This sequence reports the state of connection between network nodes SSCP1A and SSCP2A through APPN TG 21.

Second sequence: (v1,v2,e1) APPN data

```
vertex 1 :   NETA;SSCP1A

                Class   : appnNN

vertex 2 :   NETA;SSCP2A

                Class   : appnNN

endpoint 1 : NETA;SSCP1A;21.NETA.SSCP2A

                Class   : appnTransmissionGroup
                States  :
                    Operational State    : Disabled
                    Usage State          : Idle
                    Administrative State : Unlocked
                    Availability Status  : No Status
                    Procedural Status    : No Status
                    Unknown Status       : False
                    Native Status        : N/A
                Info :
                    resourceSequenceNumber        : 6
                    appnTGcapabilities            : 00
                    cp-cpSessionSupport           : TRUE
```

This sequence reports the state of connection between network nodes SSCP1A and SSCP2A through APPN TG 21. Note that the TG operational state is now disabled.

Requesting and monitoring local topology (snaLocalTopo)

This section contains the following topics:

Overview

This section describes the actions that are used to monitor the resources owned by a VTAM topology agent host.

Management of VTAM resources requires that a manager application program be able to obtain an initial list of the resources, their status, their connectivity, and selected other pertinent data to be followed later by notification of changes to the status or connectivity. The manager application program can monitor these VTAM resources by sending the ACTION(*snapshot*) request to the *snaLocalTopo* object instance at the local VTAM topology agent or at a remote VTAM topology agent.

The *snaLocalTopo* *snapshot* is used to obtain information about the following resource data:

- VTAM topology agent host

  A set of attribute data associated with the object class of the VTAM node is reported. The VTAM topology agent host is always reported as a vertex 1 in the *snapshot* response.
- Remote VTAM host

  If a remote VTAM host is connected to the VTAM topology agent host, the remote VTAM host is reported in the *snaLocalTopo* *snapshot*. The remote VTAM host is reported as a vertex 2 in the *snapshot* response.
- Owned NCPs

  For NCPs that are owned by the VTAM topology agent host, a set of attribute data associated with the NCP itself is reported. An owned NCP is reported as a *t4Node* object in vertex 2 (associated with a VTAM host vertex 1) and also is reported as a vertex 1 in the *snapshot* response.
- Contacted NCPs

  A contacted NCP is reported as a *t4Node* object in vertex 2 (associated with a local NCP vertex 1).
- Virtual routing nodes

  A virtual node is reported as a *virtualRoutingNode* object in vertex 2.
- Other contacted nodes

  Other nodes that are contacted are reported as *t2-1Node*, *lenNode*, *appnEN*, and *appnNN* objects in vertex 2. Type 1 and type 2.0 nodes are not reported in vertex 2.
- Lines

  The lines and channels attached to the VTAM topology agent host are reported; also, for any NCP reported as a vertex 1, the predefined and dynamically defined lines are reported. All lines and channels are reported as *port* objects in vertex 1.
- Physical units and link stations

  The dynamic, leased, and switched PUs and link stations associated with the VTAM topology agent host or with owned NCPs (reported as vertex 1) are reported as *logicalLink* objects in endpoint 1. Remote link stations, for nodes reported in vertex 2, are not reported as objects; instead, they are reported in the *partnerConnection* attribute of the *logicalLink* object in endpoint 1.
- Connections to adjacent nodes

  The connections from the VTAM host to adjacent nodes are reported; also, the connections from owned NCPs are reported. The connections are reported as *appnTransmissionGroup* and *subareaTransmissionGroup* objects in endpoint 1.

The VTAM topology agent host is reported as an instance of one of the following object classes:

- Type 5 node (*t5Node*)
- APPN network node (*appnNN*)

- Interchange node (*interchangeNode*)
- APPN end node (*appnEN*)
- Migration data host (*migrationDataHost*)

When a connection to an adjacent, contacted node is reported, the adjacent node is reported as an instance of one of the following object classes:
- Type 5 node (*t5Node*)
- APPN network node (*appnNN*)
- Interchange node (*interchangeNode*)
- APPN end node (*appnEN*)
- Type 4 node (*t4Node*)
- LEN node (*lenNode*)
- Type 2.1 node (*t2-1Node*)
- Virtual routing node (*virtualRoutingNode*)

A connection to an adjacent node is reported by the following resources:
- APPN transmission groups (*appnTransmissionGroup*)
- Subarea transmission groups (*subareaTransmissionGroup*)
- Lines and channels (*port*)
- Physical units (*logicalLink*)
- Link stations (*logicalLink*)

Other predefined resources (that are not currently being used for connections) are also reported as part of the *snaLocalTopo* *snapshot*.

The data reported for a *snaLocalTopo* *snapshot* can be partially controlled with the VTAMTOPO filtering option. Appendix G, "VTAMTOPO filtering option reporting," on page 347 shows a summary of the results of using the VTAMTOPO filtering option for connected switched PUs. See the *z/OS Communications Server: SNA Resource Definition Reference* and the *z/OS Communications Server: SNA Operation* for more information about the VTAMTOPO filtering option.

Action request

A *snapshot* action request is used to request local topology data from a VTAM topology agent host. The action request is sent as an m-Action-Confirmed operation.

The manager application program can request that any future updates to the *snaLocalTopo* object be returned, as they occur. The local topology data is requested without updates by specifying the `oneTimeOnly` value in the `actionInfoArg` portion of the request. The local topology data and future updates can be requested by specifying the `ongoing` value in the `actionInfoArg` portion of the request.

The target object of the request is the only object instance in the *snaLocalTopo* object class. Therefore, a single object instance name **must** be specified as the baseManagedObjectInstance in the request.

Optionally, the manager application program can specify the *appnPlusSubareaParm* parameter that indicates whether the requested data is `appnOnly` or `appnPlusSubarea`. An `appnOnly` request does not result in only APPN objects being reported. However, it does result in no NCP objects being reported. Instead, an NCP is considered part of a composite node with the VTAM topology agent host. The `appnPlusSubarea` request results in NCPs being reported as type 4 nodes. If the *appnPlusSubareaParm* is not specified in the request, the default value is `appnOnly`.

The following example shows a start request for *snaLocalTopo* data:

```
          msg CMIP-1.ROIVapdu (invokeID 196612,
operation-value 7, argument (baseManagedObjectCl
ass 1.3.18.0.0.2152, baseManagedObjectInstance (
distinguishedName (RelativeDistinguishedName (At
tributeValueAssertion (attributeType 1.3.18.0.2.
4.6, attributeValue "NETA")), RelativeDistinguis
hedName (AttributeValueAssertion (attributeType
1.3.18.0.0.2032, attributeValue "SSCP1A")), Rela
tiveDistinguishedName (AttributeValueAssertion (
attributeType 1.3.18.0.0.2216, attributeValue (s
tring "SnaLocalTopology"))))), actionInfo (actio
nType 1.3.18.0.0.2222, actionInfoArg (start ongo
ing, addlInfo (ManagementExtension (identifier 1
.3.18.0.0.2164, significance TRUE, information 1
))))))
```

Note from the example that the *snaLocalTopo* object name is composed of the network identifier and node name of the VTAM topology agent host, followed by the *graphId (1.3.18.0.0.2216)* naming attribute, which is required to have a value of `(string "SnaLocalTopology")`. The actionType is a *snapshot*, and the `actionInfoArg` indicates that this request is to start an `ongoing` *snapshot*. Also in this example the *appnPlusSubareaParm* (1.3.18.0.0.2164) is specified with information value of 1, which means `appnPlusSubarea`. An information value of 0 means `appnOnly`. The significance value of `TRUE` means that if the VTAM topology agent finds an error associated with the specification of this *appnPlusSubareaParm* parameter, then the manager application program needs to receive an error response (ROER). A `FALSE` value tells the VTAM topology agent to ignore the parameter if an error is found.

This is a typical example of a *snaLocalTopo* *snapshot* request that can be used simply by supplying the appropriate network ID and SSCP or CP name of the VTAM topology agent host.

Initial data response

Both `ongoing` and `oneTimeOnly` *snapshot*s receive a set of initial data as part of the *snapshot* response. The initial data is the report of the appropriate resource data and connectivity as it exists at the time the request is processed. The contents of the initial data depend most on the current configuration of the local resources and somewhat on the network configuration. The contents of the initial data also depends on the value of the *appnPlusSubareaParm* specified in the *snapshot* request. If the *appnPlusSubareaParm* value is 0 (`appnOnly`), the NCPs are not reported separately from the VTAM host; they are considered part of the VTAM composite node.

The initial data is sent by a set of linked-reply messages; each linked-reply message contains one or more sets of the sequence (vertex 1, vertex 2, endpoint 1), abbreviated as (v1,v2,e1). Vertex 1 represents either the VTAM topology agent host node or an owned NCP node. Vertex 1 optionally contains *port* information. Vertex 2 represents a contacted node adjacent to the node specified in the associated vertex 1; vertex 2 is optional. If both vertex 1 and vertex 2 are present, endpoint 1 represents the transmission group used for the connection and the *logicalLink* on the vertex 1 side of the connection. If only vertex 1 is present (no vertex 2), endpoint 1 represents a *logicalLink* at the vertex 1 node that is not currently being used for a connection. Endpoint 1 is optional.

The first time a node is reported as vertex 1 in a *snaLocalTopo* *snapshot*, the full set of attribute data associated with that node is reported; this is referred to as the

long form of vertex 1. For subsequent reports of that same node in vertex 1, the attribute data is omitted; this is called the **short form** of vertex 1.

Each sequence of (v1,v2,e1) can report at most one *port*, one *logicalLink*, one transmission group, and two nodes. All resources reported in a single sequence of (v1,v2,e1) are related.

To report other VTAM resources, such as other lines or PUs, VTAM must be repeated as a vertex 1 for each sequence of (v1,v2,e1) until the list of resources is exhausted. When all initial data has been sent, the VTAM topology agent sends one additional linked-reply message; the value in the `actionReplyInfo` field of this message is '()', an empty set.

If the *snaLocalTopo snapshot* request is the `oneTimeOnly` type, the empty set linked-reply is followed by the final *snapshot* RORS response message. If the *snapshot* is the `ongoing` type, no additional messages are sent by the VTAM topology agent until a reportable change occurs to a resource within the scope of the *snaLocalTopo snapshot* request.

Update data response

For `ongoing` *snapshot*s after the initial data has been sent, all subsequent reportable resource changes are reported in linked-reply messages. The same message syntax is used for the update data as is used for initial data. In fact, a single linked-reply message does not indicate whether the data is initial data or update data; it depends solely on whether the message arrived before the empty set linked-reply or after it.

Update data is generated by the VTAM topology agent for these reasons:

- An object is created.

  Objects are reported when VTAM learns about them. For example, when a major node is activated, VTAM learns about the resources defined in the major node. When a connection is established, VTAM learns about the contacted node and reports the node. The long form of the object is always reported for an object creation, with the `reason` field omitted, implying the default value of `addOrUpdate`.
- An object is deleted.

  Objects are reported as deleted when the object definition is deleted from VTAM, as happens when a major node is inactivated. Objects are also reported as deleted when they logically cease to exist, as is the case with an APPN transmission group whose connection has been broken. The short form of an object is reported for an object deletion, with `reason` value of `deleted`.
- An object changes state.

  An object is reported when the internal VTAM state of the object changes, such as from pending-active to active. The internal VTAM state is mapped to the extended OSI state attributes, which are reported. Because several VTAM internal states are mapped to the same set of OSI state attribute values, it is possible for the internal VTAM status to change but the derived OSI states not to change. These unchanged OSI states might still be reported by VTAM. Note that it is likely that most of the resource state changes will not each result in a sequence of (v1,v2,e1) being sent since the merge process holds and merges updates where the resources are in non-resting states. For more information about the merge process, refer to "ACTION(snapshot) update merging" on page 168.

The short form of the object is generally reported for state changes; however, important attributes may also be reported with the state changes. The `reason` field is omitted, implying the default value of `addOrUpdate`.

- An attribute value changes.

  An attribute value change refers to changes in resource data other than the state of the resource. In general, the VTAM topology agent does not support the reporting of attribute value changes; however, there are instances of attribute value changes that are considered to be too important to ignore. These few selected changes are reported. The short form of an object is reported for an attribute value change; however, there will always be a small set of attributes also reported. The `reason` field is omitted, implying the default value of `addOrUpdate`.

Table 13 shows the *snaLocalTopo* update data and the reasons for the updates.

Note: MODIFY VTAMTOPO can generate a snaLocalTopo Update.

Table 13. Resources with reason for snaLocalTopo update data

Resource	Reason	Notes®
Local VTAM	None	Local VTAM is never the cause of update data
NCP	Vertex 1 created	NCP major node activated
	Vertex 1 deleted	NCP major node deactivated
	Vertex 1 state change	NCP changed state
	Vertex 1 attribute value change	Learned gateway information; report attributes *gatewayNode* or *interconnectedNetIds* or both
Line	Created	Major node containing line activated
	Deleted	Major node containing line deactivated
	State change	Line changed state
	Attribute value change	Report learned data in attributes: *adapterAddresses* or *relatedAdapter*.
PU or link station	Created	Major node containing PU activated
	Deleted	Major node containing PU deactivated
	State change	PU changed state
	Attribute value change	If associated LINE information changes, report attributes *portId* or *adjacentLinkStationAddress* or both
Contacted node	Created	New connection established to node
	Deleted	Loss of connection to node
Transmission group	Created	New connection established
	Deleted	Loss of a connection

The VTAM topology agent continues to send update data until a valid request is received to stop the *snapshot*.

Action termination

The VTAM topology agent terminates an `ongoing` *snaLocalTopo* *snapshot* action under the following conditions:

- A stop *snapshot* action request is received.
- An error occurs during *snapshot* processing in VTAM.
- The association between the local CMIP services and the manager application program's CMIP services terminates.

The following is an example of a valid stop request:

```
           msg CMIP-1.ROIVapdu (invokeID 196613,
operation-value 7, argument (baseManagedObjectCla
ss 1.3.18.0.0.2152, baseManagedObjectInstance (di
stinguishedName (RelativeDistinguishedName (Attri
buteValueAssertion (attributeType 1.3.18.0.2.4.6,
 attributeValue "NETA")), RelativeDistinguishedNa
me (AttributeValueAssertion (attributeType 1.3.18
.0.0.2032, attributeValue "SSCP1A")), RelativeDis
tinguishedName (AttributeValueAssertion (attribut
eType 1.3.18.0.0.2216, attributeValue (string "Sn
aLocalTopology"))))), actionInfo (actionType 1.3.
18.0.0.2222, actionInfoArg (stop 196612))))
```

The stop request in the previous example looks very similar to the associated start request except for the `actionInfoArg` portion of the request. For the stop request, the `stop` keyword is used along with the invoke identifier of the start request that is to be terminated.

The result of the VTAM topology agent's processing of a stop request is three messages:
- An empty set linked-reply for the start request
- An RORS response to the start request
- An RORS response to the stop request

The following shows examples of these three messages:

First, the empty set linked-reply: (Note that the associated invoke identifier in a linked-reply is given in the `linked-ID` field.)

```
                   msg CMIP-1.ROIVapdu (invokeID
 1, linked-ID 196612, operation-value 2, argument
 (actionResult (managedObjectClass 1.3.18.0.0.215
2, managedObjectInstance (distinguishedName (Rela
tiveDistinguishedName (AttributeValueAssertion (a
ttributeType 1.3.18.0.2.4.6, attributeValue "NETA
")), RelativeDistinguishedName (AttributeValueAss
ertion (attributeType 1.3.18.0.0.2032, attributeV
alue "SSCP1A")), RelativeDistinguishedName (Attri
buteValueAssertion (attributeType 1.3.18.0.0.2216
, attributeValue (string "SnaLocalTopology"))))),
 actionReply (actionType 1.3.18.0.0.2222, actionR
eplyInfo ())))
```

Next, the RORS for the start request:

```
                  msg CMIP-1.RORSapdu (invokeID
196612)
```

Finally, the RORS for the stop request:

```
                   msg CMIP-1.RORSapdu (invokeID
 196613, resultOption (operation-value 7, result
(managedObjectClass 1.3.18.0.0.2152, managedObjec
tInstance (distinguishedName (RelativeDistinguish
edName (AttributeValueAssertion (attributeType 1.
3.18.0.2.4.6, attributeValue "NETA")), RelativeDi
stinguishedName (AttributeValueAssertion (attribu
```

```
teType 1.3.18.0.0.2032, attributeValue "SSCP1A"))
, RelativeDistinguishedName (AttributeValueAssert
ion (attributeType 1.3.18.0.0.2216, attributeValu
e (string "SnaLocalTopology")))))), actionReply (a
ctionType 1.3.18.0.0.2222, actionReplyInfo ())))))
```

snaLocalTopo snapshot data

For local topology data, the structure of the **long form** of vertex 1 is defined as follows:

vertex1

Identifies the beginning of vertex 1. The lower level fields in vertex 1 are:

object Object instance name of the node represented by vertex 1. VTAM always returns the `distinguishedName` form of an object instance name.

class Object identifier (OI) representing the object class of the reported node.

states String of characters representing the OSI state of the vertex 1 node. For a list of OSI states, refer to "OSI object states" on page 155.

info Set of attributes providing data about the vertex 1 node. Not all attributes appear in all instances of vertex 1. The list of all possible attributes is:

dependencies
Included only if vertex 1 represents an NCP.

opEquipmentList
Included only if vertex 1 represents the local VTAM host.

softwareList
Included only if vertex 1 represents the local VTAM host.

sysplexInfo
Included only if vertex 1 represents the local VTAM host.

appnNodeCapabilities
Included only if vertex 1 represents the local VTAM host and the VTAM node is an *appnNN* or an *interchangeNode*.

extendedAppnNodeCapabilities
Included only if vertex 1 represents the local VTAM host and the VTAM node is an *appnNN* or an *interchangeNode*.

subareaLimit
Included only if vertex 1 represents a subarea node, which is a *t5Node*, *t4Node*, *interchangeNode*, or *migrationDataHost*.

subareaAddress
Included only if vertex 1 represents a subarea node, which is a *t5Node*, *t4Node*, *interchangeNode*, or *migrationDataHost*.

puName
Included only if vertex 1 represents the local VTAM host, and VTAM is subarea-capable.

gatewayNode
Included only if vertex 1 represents an NCP, and the NCP is gateway-capable.

gatewaySSCP
Included only if vertex 1 represents the local VTAM host, and VTAM is gateway-capable.

interconnectedNetIds
Included only if vertex 1 represents an NCP, and the NCP is gateway-capable.

moreInfo
Included only if a *port* object is to be reported in vertex 1 and is followed by a ManagementExtension that contains the *port* object.

identifier
Object identifier for the *objectStuffInMoreInfoParm*, which identifies the syntax for this ManagementExtension.

information
Port data in the ManagementExtension, which contains the following fields:

object Object instance name of the *port* to be reported. VTAM always reports the `distinguishedName` form of object instance name.

class Object identifier (OI) representing the *port* object class.

states String of characters representing the OSI state of the *port*. For a list of OSI states, refer to "OSI object states" on page 155.

info Set of attributes providing data about the *port*. Not all attributes appear in all instances of *port* objects. The list of all possible attributes is:

dependencies
Always included.

adapterNumbers
Included only if the *snapshot* type is `appnOnly`.

connectionId
Included only if the *snapshot* type is `appnPlusSubarea`.

adapterAddresses
Always included.

lineType
Always included.

dlcName
Always included.

relatedAdapter
Always included.

The structure of the **short form** of vertex 1 is defined as follows:

vertex1
Identifies the beginning of vertex 1. The lower level fields in vertex 1 are:

object Object instance name of the node represented by vertex 1. VTAM always returns the `distinguishedName` form of object instance name.

class Object identifier (OI) representing the object class of the reported node.

states String of characters representing the OSI state of the vertex 1 node only if a state change occurred for the vertex 1 node. For a list of OSI states, refer to "OSI object states" on page 155.

info Set of attributes providing data about the vertex 1 node. Not all attributes appear in all instances of vertex 1. The list of all possible attributes is:

gatewayNode
Included only if vertex 1 represents an NCP and VTAM has discovered that the NCP is gateway-capable.

interconnectedNetIds
Included only if vertex 1 represents an NCP and VTAM has discovered that the NCP is gateway-capable.

moreInfo
Included only if a *port* object is to be reported in vertex 1 and is followed by a ManagementExtension that contains the *port* object. The form included here is the short form of the *port*.

identifier
Object identifier for the *objectStuffInMoreInfoParm*, which identifies the syntax for this ManagementExtension.

information
Port data in the ManagementExtension, which contains the following fields:

object Object instance name of the *port* to be reported. VTAM always reports the `distinguishedName` form of object instance name.

class Object identifier (OI) representing the *port* object class.

states String of characters representing the OSI state of the *port*. For a list of OSI states, refer to "OSI object states" on page 155.

info Set of attributes providing data about the *port*. Not all attributes appear in all instances of *port* objects. The list of all possible attributes is:

adapterAddresses
Included only to report an attribute value change.

relatedAdapter
Included only to report an attribute value change.

reason Included only if the *port* is reported as `deleted`.

reason Included only if the vertex 1 node is being reported as `deleted`.

Note that although the previous lists show the long form of the *port* in the long form of vertex 1 and the short form of the *port* in the short form of vertex 1, it is possible to have the long form of *port* contained in the short form of vertex 1. For example, if a node was already reported in a previous vertex 1 (long form) and was reported again (short form), but this time the vertex 1 included a *port* that has not yet been reported, the *port* is the long form. The VTAM topology agent never reports the short form of a *port* in a long form of vertex 1.

Vertex 2 for *snaLocalTopo* does not have a long form and a short form; there is only one form. The structure of vertex 2 is as follows:

vertex2
Identify the beginning of vertex 2. The lower level fields in vertex 2 are:

object Object instance name of the node represented by vertex 2. VTAM always returns the `distinguishedName` form of object instance name.

class Object identifier (OI) representing the object class of the reported node.

VTAM might not report the true object class of composite nodes such as interchange nodes. When reporting subarea connections, vertex2 object class is always *t5Node* or *t4Node*.

reason Included only if the vertex 2 node is being reported as `deleted`.

The structure of endpoint 1 for *snaLocalTopo* is variable, depending on what objects are reported. Endpoint 1 can contain both a transmission group object and a *logicalLink* object, or it can contain just a *logicalLink* object. Endpoint 1 never contains only a transmission group object. If both a TG and a *logicalLink* are reported, the TG is reported as the first and primary object in endpoint 1. If only a *logicalLink* is reported, the *logicalLink* is reported as the primary object in endpoint 1.

The structure of endpoint 1 is shown in two parts; first, the structure of the TG object is shown, including where the *logicalLink* object fits into the structure. Then the *logicalLink* object is shown with a structure used for either the primary or secondary object in endpoint 1.

The **TG** object structure follows:

object Object instance name of the TG to be reported. VTAM always reports the `distinguishedName` form of object instance name.

class Object identifier (OI) representing either the *appnTransmissionGroup* or the *subareaTransmissionGroup* object class.

states String of characters representing the OSI state of the TG; this field is included only for *appnTransmissionGroup* objects. For a list of OSI states, refer to "OSI object states" on page 155.

info Set of attributes providing data about the TG. Not all attributes appear in all instances of TG objects. The info label and value are included only for *appnTransmissionGroup* objects. The list of all possible attributes is:

cp-cpSessionSupport
Included only for *appnTransmissionGroup* objects.

appnTGCapabilities
Included only for *appnTransmissionGroup* objects.

moreInfo
Always included for a TG object and is followed by a ManagementExtension that contains the *logicalLink* object.

identifier
Object identifier for the *objectStuffInMoreInfoParm*, which is a parameter that identifies the syntax for this ManagementExtension.

information
logicalLink data in the ManagementExtension. At this point the *logicalLink* data described below is inserted.

reason Included only if the TG is being reported as `deleted`.

The **logical link** object structure follows:

object Object instance name of the *logicalLink* to be reported. VTAM always reports the `distinguishedName` form of object instance name.

class Object identifier (OI) representing the *logicalLink* object class.

states String of characters representing the OSI state of the *logicalLink*. For a list of OSI states, refer to "OSI object states" on page 155.

info Set of attributes providing data about the *logicalLink*. Not all attributes appear in all instances of *logicalLink* objects. The list of all possible attributes is:

dependencies
Included for initial data, object creation updates, state change updates for switched PUs, and attribute value change updates caused by line filtering through the MODIFY VTAMTOPO command.

connectionId
Included only for native ATM connections and only if the *snapshot* type is `appnPlusSubarea`, for initial data and object creation updates.

portId Reported for initial data, object creation updates, state change updates for switched PUs, and selected attribute value change updates; however, this attribute might be suppressed if it refers to a switched logical line and switched logical lines are being suppressed (because of how the VTAMTOPO filtering option is specified).

partnerConnection
Included only if the resource represented by this *logicalLink* is a link station and if the partner *logicalLink* information is available. Included for initial data, object creation updates, and state change updates.

adjacentLinkStationAddress
Included if the associated line is a logical token ring line, a frame relay line, an XCA line, an XCF line, an ATM line, or an SDLC line with a polling address. The attribute might also be included for PUs that have the ADDR keyword coded on the PU definition statement. The attribute is included for initial data, object creation updates, state change updates, and selected attribute value change updates.

adjacentNodeType
Included for initial data, object creation updates, and state change updates.

dlurLocalLsAddress
Included when a DLUR supports downstream PUs. *LogicalLink* reports local addressing information.

dlurName
Included only if the PU is attached to VTAM through the dependent LU server and dependent LU requester capabilities. Included for initial data, object creation updates, and state change updates.

reason Included only if the *logicalLink* is being reported as `deleted`.

snaLocalTopo snapshot example

The following example of *snaLocalTopo* *snapshot* response data shows only the first linked-reply message of the initial data; the sample configuration is the VTAM topology agent host (SSCP1A) connected to an active NCP (NCP3AB8). After the example string, the contents of the message are described in further detail.

```
                    msg CMIP-1.ROIVapdu (invokeID
 131074, linked-ID 196612, operation-value 2, arg
ument (actionResult (managedObjectClass 1.3.18.0.
0.2152, managedObjectInstance (distinguishedName
(RelativeDistinguishedName (AttributeValueAsserti
on (attributeType 1.3.18.0.2.4.6, attributeValue
"NETA")), RelativeDistinguishedName (AttributeVal
ueAssertion (attributeType 1.3.18.0.0.2032, attri
buteValue "SSCP1A")), RelativeDistinguishedName (
AttributeValueAssertion (attributeType 1.3.18.0.0
.2216, attributeValue (string "SnaLocalTopology")
)))), actionReply (actionType 1.3.18.0.0.2222, ac
tionReplyInfo ((vertex1 (object (distinguishedNam
e (RelativeDistinguishedName (AttributeValueAsser
tion (attributeType 1.3.18.0.2.4.6, attributeValu
e "NETA")), RelativeDistinguishedName (AttributeV
alueAssertion (attributeType 1.3.18.0.0.2032, att
ributeValue "SSCP1A")))), class 1.3.18.0.0.1826,
states 01010100000000, info (Attribute (attribute
Id 1.3.14.2.2.4.33, attributeValue (ObjectInstanc
e (distinguishedName (RelativeDistinguishedName (
AttributeValueAssertion (attributeType 1.3.18.0.2
.4.8, attributeValue "ORGREG")), RelativeDistingu
ishedName (AttributeValueAssertion (attributeType
 2.5.4.10, attributeValue "IBM")), RelativeDistin
guishedName (AttributeValueAssertion (attributeTy
pe 1.3.14.2.2.4.45, attributeValue "9021")), Rela
tiveDistinguishedName (AttributeValueAssertion (a
ttributeType 1.3.14.2.2.4.50, attributeValue "032
082")))))), Attribute (attributeId 1.3.14.2.2.4.5
3, attributeValue (ObjectInstance (distinguishedN
ame (RelativeDistinguishedName (AttributeValueAss
ertion (attributeType 1.3.18.0.2.4.8, attributeVa
lue "ORGREG")), RelativeDistinguishedName (Attrib
uteValueAssertion (attributeType 2.5.4.10, attrib
uteValue "IBM")), RelativeDistinguishedName (Attr
ibuteValueAssertion (attributeType 0.0.13.3100.0.
7.38, attributeValue (pString "ACF/VTAM.4.3.0")))
)))), Attribute (attributeId 1.3.18.0.0.2296, att
ributeValue "SYSPLEX"), Attribute (attributeId 1.
3.18.0.0.1940, attributeValue 3340), Attribute (a
ttributeId 1.3.18.0.0.1970, attributeValue 0000),
 Attribute (attributeId 1.3.18.0.0.2036, attribut
```

```
eValue 511), Attribute (attributeId 1.3.18.0.0.20
35, attributeValue 1), Attribute (attributeId 1.3
.18.0.0.2013, attributeValue "ISTPUS"), Attribute
 (attributeId 1.3.18.0.0.1972, attributeValue TRU
E)), moreInfo (ManagementExtension (identifier 1.
3.18.0.0.2162, information (object (distinguished
Name (RelativeDistinguishedName (AttributeValueAs
sertion (attributeType 1.3.18.0.2.4.6, attributeV
alue "NETA")), RelativeDistinguishedName (Attribu
teValueAssertion (attributeType 1.3.18.0.0.2032,
attributeValue "SSCP1A")), RelativeDistinguishedN
ame (AttributeValueAssertion (attributeType 1.3.1
8.0.0.2142, attributeValue "0321-L")))), class 1.
3.18.0.0.2089, states 01010100000000, info (Attri
bute (attributeId 1.3.18.0.0.2194, attributeValue
 (dependents (and (Dependents (item (distinguishe
dName (RelativeDistinguishedName (AttributeValueA
ssertion (attributeType 1.3.18.0.2.4.6, attribute
Value "NETA")), RelativeDistinguishedName (Attrib
uteValueAssertion (attributeType 1.3.18.0.0.2032,
 attributeValue "SSCP1A")), RelativeDistinguished
Name (AttributeValueAssertion (attributeType 1.3.
18.0.0.2272, attributeValue "NCP.ISTPUS")))))), De
pendents (item (distinguishedName (RelativeDistin
guishedName (AttributeValueAssertion (attributeTy
pe 1.3.18.0.2.4.6, attributeValue "NETA")), Relat
iveDistinguishedName (AttributeValueAssertion (at
tributeType 1.3.18.0.0.2032, attributeValue "SSCP
1A")))))))), Attribute (attributeId 2.9.3.5.7.1,
 attributeValue "0321"), Attribute (attributeId 1
.3.18.0.0.2117, attributeValue ()), Attribute (at
tributeId 1.3.18.0.0.2131, attributeValue nonswit
ched), Attribute (attributeId 1.3.18.0.0.2127, at
tributeValue "CHANNEL"), Attribute (attributeId 1
.3.18.0.0.2244, attributeValue (noInfo NULL))))))
), vertex2 (object (distinguishedName (RelativeDi
stinguishedName (AttributeValueAssertion (attribu
teType 1.3.18.0.2.4.6, attributeValue "NETA")), R
elativeDistinguishedName (AttributeValueAssertion
 (attributeType 1.3.18.0.0.2032, attributeValue "
NCP3AB8")))), class 1.3.18.0.0.1844), endpoint1 (
object (distinguishedName (RelativeDistinguishedN
ame (AttributeValueAssertion (attributeType 1.3.1
8.0.2.4.6, attributeValue "NETA")), RelativeDisti
nguishedName (AttributeValueAssertion (attributeT
ype 1.3.18.0.0.2032, attributeValue "SSCP1A")), R
elativeDistinguishedName (AttributeValueAssertion
 (attributeType 1.3.18.0.0.2044, attributeValue "
1.1.NETA.3.NCP3AB8")))), class 1.3.18.0.0.1840, m
oreInfo (ManagementExtension (identifier 1.3.18.0
.0.2162, information (object (distinguishedName (
RelativeDistinguishedName (AttributeValueAssertio
n (attributeType 1.3.18.0.2.4.6, attributeValue "
NETA")), RelativeDistinguishedName (AttributeValu
eAssertion (attributeType 1.3.18.0.0.2032, attrib
uteValue "SSCP1A")), RelativeDistinguishedName (A
ttributeValueAssertion (attributeType 1.3.18.0.0.
2133, attributeValue "0321-S")))), class 1.3.18.0
.0.2085, states 01010100000000, info (Attribute (
attributeId 1.3.18.0.0.2194, attributeValue (depe
ndents (and (Dependents (item (distinguishedName
(RelativeDistinguishedName (AttributeValueAsserti
on (attributeType 1.3.18.0.2.4.6, attributeValue
"NETA")), RelativeDistinguishedName (AttributeVal
ueAssertion (attributeType 1.3.18.0.0.2032, attri
buteValue "SSCP1A")), RelativeDistinguishedName (
AttributeValueAssertion (attributeType 1.3.18.0.0
```

```
.2142, attributeValue "0321-L")))))))), Attribut
e (attributeId 1.3.18.0.0.2142, attributeValue "0
321-L"), Attribute (attributeId 1.3.18.0.0.2236,
attributeValue (object (distinguishedName (Relati
veDistinguishedName (AttributeValueAssertion (att
ributeType 1.3.18.0.2.4.6, attributeValue "NETA")
), RelativeDistinguishedName (AttributeValueAsser
tion (attributeType 1.3.18.0.0.2032, attributeVal
ue "NCP3AB8")), RelativeDistinguishedName (Attrib
uteValueAssertion (attributeType 1.3.18.0.0.2133,
 attributeValue "PU321A")))))), Attribute (attrib
uteId 1.3.18.0.0.2121, attributeValue t4))))))),
(vertex1 (object (distinguishedName (RelativeDist
inguishedName (AttributeValueAssertion (attribute
Type 1.3.18.0.2.4.6, attributeValue "NETA")), Rel
ativeDistinguishedName (AttributeValueAssertion (
attributeType 1.3.18.0.0.2032, attributeValue "SS
CP1A")), RelativeDistinguishedName (AttributeValu
eAssertion (attributeType 1.3.18.0.0.2032, attrib
uteValue "NCP3AB8")))), class 1.3.18.0.0.1844, st
ates 01010100000000, info (Attribute (attributeId
 1.3.18.0.0.2194, attributeValue (dependents (and
 (Dependents (item (distinguishedName (RelativeDi
stinguishedName (AttributeValueAssertion (attribu
teType 1.3.18.0.2.4.6, attributeValue "NETA")), R
elativeDistinguishedName (AttributeValueAssertion
 (attributeType 1.3.18.0.0.2032, attributeValue "
SSCP1A")), RelativeDistinguishedName (AttributeVa
lueAssertion (attributeType 1.3.18.0.0.2272, attr
ibuteValue "NCP.NCP3AB8"))))), Dependents (item (
distinguishedName (RelativeDistinguishedName (Att
ributeValueAssertion (attributeType 1.3.18.0.2.4.
6, attributeValue "NETA")), RelativeDistinguished
Name (AttributeValueAssertion (attributeType 1.3.
18.0.0.2032, attributeValue "SSCP1A")))))))), At
tribute (attributeId 1.3.18.0.0.2036, attributeVa
lue 255), Attribute (attributeId 1.3.18.0.0.2035,
 attributeValue 3)), moreInfo (ManagementExtensio
n (identifier 1.3.18.0.0.2162, information (objec
t (distinguishedName (RelativeDistinguishedName (
AttributeValueAssertion (attributeType 1.3.18.0.2
.4.6, attributeValue "NETA")), RelativeDistinguis
hedName (AttributeValueAssertion (attributeType 1
.3.18.0.0.2032, attributeValue "SSCP1A")), Relati
veDistinguishedName (AttributeValueAssertion (att
ributeType 1.3.18.0.0.2142, attributeValue "LN3A6
")))), class 1.3.18.0.0.2089, states 010101000000
00, info (Attribute (attributeId 1.3.18.0.0.2194,
 attributeValue (dependents (and (Dependents (ite
m (distinguishedName (RelativeDistinguishedName (
AttributeValueAssertion (attributeType 1.3.18.0.2
.4.6, attributeValue "NETA")), RelativeDistinguis
hedName (AttributeValueAssertion (attributeType 1
.3.18.0.0.2032, attributeValue "SSCP1A")), Relati
veDistinguishedName (AttributeValueAssertion (att
ributeType 1.3.18.0.0.2272, attributeValue "NCP.N
CP3AB8"))))), Dependents (item (distinguishedName
 (RelativeDistinguishedName (AttributeValueAssert
ion (attributeType 1.3.18.0.2.4.6, attributeValue
 "NETA")), RelativeDistinguishedName (AttributeVa
lueAssertion (attributeType 1.3.18.0.0.2032, attr
ibuteValue "SSCP1A")), RelativeDistinguishedName
(AttributeValueAssertion (attributeType 1.3.18.0.
0.0.2032, attributeValue "NCP3AB8")))))))), Attrib
ute (attributeId 2.9.3.5.7.1, attributeValue "030
3"), Attribute (attributeId 1.3.18.0.0.2117, attr
ibuteValue ()), Attribute (attributeId 1.3.18.0.0
```

```
.2131, attributeValue nonswitched), Attribute (at
tributeId 1.3.18.0.0.2127, attributeValue "SDLC")
, Attribute (attributeId 1.3.18.0.0.2244, attribu
teValue (noInfo NULL))))))), endpoint1 (object (d
istinguishedName (RelativeDistinguishedName (Attr
ibuteValueAssertion (attributeType 1.3.18.0.2.4.6
, attributeValue "NETA")), RelativeDistinguishedN
ame (AttributeValueAssertion (attributeType 1.3.1
8.0.0.2032, attributeValue "SSCP1A")), RelativeDi
stinguishedName (AttributeValueAssertion (attribu
teType 1.3.18.0.0.2133, attributeValue "P3A3274E"
)))), class 1.3.18.0.0.2085, states 0101010000000
0, info (Attribute (attributeId 1.3.18.0.0.2194,
attributeValue (dependents (and (Dependents (item
 (distinguishedName (RelativeDistinguishedName (A
ttributeValueAssertion (attributeType 1.3.18.0.2.
4.6, attributeValue "NETA")), RelativeDistinguish
edName (AttributeValueAssertion (attributeType 1.
3.18.0.0.2032, attributeValue "SSCP1A")), Relativ
eDistinguishedName (AttributeValueAssertion (attr
ibuteType 1.3.18.0.0.2272, attributeValue "NCP.NC
P3AB8"))))), Dependents (item (distinguishedName
(RelativeDistinguishedName (AttributeValueAsserti
on (attributeType 1.3.18.0.2.4.6, attributeValue
"NETA")), RelativeDistinguishedName (AttributeVal
ueAssertion (attributeType 1.3.18.0.0.2032, attri
buteValue "SSCP1A")), RelativeDistinguishedName (
AttributeValueAssertion (attributeType 1.3.18.0.0
.2142, attributeValue "LN3A6"))))))))), Attribute
 (attributeId 1.3.18.0.0.2142, attributeValue "LN
3A6"), Attribute (attributeId 1.3.18.0.0.2236, at
tributeValue (noInfo NULL)), Attribute (attribute
Id 1.3.18.0.0.2119, attributeValue (lsAddr C2)),
Attribute (attributeId 1.3.18.0.0.2121, attribute
Value t20)))), (vertex1 (object (distinguishedNam
e (RelativeDistinguishedName (AttributeValueAsser
tion (attributeType 1.3.18.0.2.4.6, attributeValu
e "NETA")), RelativeDistinguishedName (AttributeV
alueAssertion (attributeType 1.3.18.0.0.2032, att
ributeValue "SSCP1A")), RelativeDistinguishedName
 (AttributeValueAssertion (attributeType 1.3.18.0
.0.2032, attributeValue "NCP3AB8")))), class 1.3.
18.0.0.1844), vertex2 (object (distinguishedName
(RelativeDistinguishedName (AttributeValueAsserti
on (attributeType 1.3.18.0.2.4.6, attributeValue
"NETA")), RelativeDistinguishedName (AttributeVal
ueAssertion (attributeType 1.3.18.0.0.2032, attri
buteValue "P3A4956G")))), class 1.3.18.0.0.1827),
 endpoint1 (object (distinguishedName (RelativeDi
stinguishedName (AttributeValueAssertion (attribu
teType 1.3.18.0.2.4.6, attributeValue "NETA")), R
elativeDistinguishedName (AttributeValueAssertion
 (attributeType 1.3.18.0.0.2032, attributeValue "
SSCP1A")), RelativeDistinguishedName (AttributeVa
lueAssertion (attributeType 1.3.18.0.0.2044, attr
ibuteValue "0.NETA.P3A4956G")))), class 1.3.18.0.
0.1823, states 010101000000FF, info (Attribute (a
ttributeId 1.3.18.0.0.1958, attributeValue FALSE)
, Attribute (attributeId 1.3.18.0.0.1941, attribu
teValue 00)), moreInfo (ManagementExtension (iden
tifier 1.3.18.0.0.2162, information (object (dist
inguishedName (RelativeDistinguishedName (Attribu
teValueAssertion (attributeType 1.3.18.0.2.4.6, a
ttributeValue "NETA")), RelativeDistinguishedName
 (AttributeValueAssertion (attributeType 1.3.18.0
.0.2032, attributeValue "SSCP1A")), RelativeDisti
nguishedName (AttributeValueAssertion (attributeT
```

```
ype 1.3.18.0.0.2133, attributeValue "P3A4956G")))
), class 1.3.18.0.0.2085, states 01010100000000,
info (Attribute (attributeId 1.3.18.0.0.2194, att
ributeValue (dependents (and (Dependents (item (d
istinguishedName (RelativeDistinguishedName (Attr
ibuteValueAssertion (attributeType 1.3.18.0.2.4.6
, attributeValue "NETA")), RelativeDistinguishedN
ame (AttributeValueAssertion (attributeType 1.3.1
8.0.0.2032, attributeValue "SSCP1A")), RelativeDi
stinguishedName (AttributeValueAssertion (attribu
teType 1.3.18.0.0.2272, attributeValue "NCP.NCP3A
B8")))))), Dependents (item (distinguishedName (Re
lativeDistinguishedName (AttributeValueAssertion
(attributeType 1.3.18.0.2.4.6, attributeValue "NE
TA")), RelativeDistinguishedName (AttributeValueA
ssertion (attributeType 1.3.18.0.0.2032, attribut
eValue "SSCP1A")), RelativeDistinguishedName (Att
ributeValueAssertion (attributeType 1.3.18.0.0.21
42, attributeValue "LN3A6")))))))))), Attribute (a
ttributeId 1.3.18.0.0.2142, attributeValue "LN3A6
"), Attribute (attributeId 1.3.18.0.0.2236, attri
buteValue (noInfo NULL)), Attribute (attributeId
1.3.18.0.0.2119, attributeValue (lsAddr C4)), Att
ribute (attributeId 1.3.18.0.0.2121, attributeVal
ue len)))))))), (vertex1 (object (distinguishedNam
e (RelativeDistinguishedName (AttributeValueAsser
tion (attributeType 1.3.18.0.2.4.6, attributeValu
e "NETA")), RelativeDistinguishedName (AttributeV
alueAssertion (attributeType 1.3.18.0.0.2032, att
ributeValue "SSCP1A")), RelativeDistinguishedName
 (AttributeValueAssertion (attributeType 1.3.18.0
.0.2032, attributeValue "NCP3AB8")))), class 1.3.
18.0.0.1844, moreInfo (ManagementExtension (ident
ifier 1.3.18.0.0.2162, information (object (disti
nguishedName (RelativeDistinguishedName (Attribut
eValueAssertion (attributeType 1.3.18.0.2.4.6, at
tributeValue "NETA")), RelativeDistinguishedName
(AttributeValueAssertion (attributeType 1.3.18.0.
0.2032, attributeValue "SSCP1A")), RelativeDistin
guishedName (AttributeValueAssertion (attributeTy
pe 1.3.18.0.0.2142, attributeValue "LN3A1")))), c
lass 1.3.18.0.0.2089, states 01010100000000, info
 (Attribute (attributeId 1.3.18.0.0.2194, attribu
teValue (dependents (and (Dependents (item (disti
nguishedName (RelativeDistinguishedName (Attribut
eValueAssertion (attributeType 1.3.18.0.2.4.6, at
tributeValue "NETA")), RelativeDistinguishedName
(AttributeValueAssertion (attributeType 1.3.18.0.
0.2032, attributeValue "SSCP1A")), RelativeDistin
guishedName (AttributeValueAssertion (attributeTy
pe 1.3.18.0.0.2272, attributeValue "NCP.NCP3AB8")
)))), Dependents (item (distinguishedName (Relati
veDistinguishedName (AttributeValueAssertion (att
ributeType 1.3.18.0.2.4.6, attributeValue "NETA")
), RelativeDistinguishedName (AttributeValueAsser
tion (attributeType 1.3.18.0.0.2032, attributeVal
ue "SSCP1A")), RelativeDistinguishedName (Attribu
teValueAssertion (attributeType 1.3.18.0.0.2032,
attributeValue "NCP3AB8")))))))), Attribute (att
ributeId 2.9.3.5.7.1, attributeValue "0305"), Att
ribute (attributeId 1.3.18.0.0.2117, attributeVal
ue ()), Attribute (attributeId 1.3.18.0.0.2131, a
ttributeValue nonswitched), Attribute (attributeI
d 1.3.18.0.0.2127, attributeValue "SDLC"), Attrib
ute (attributeId 1.3.18.0.0.2244, attributeValue
(noInfo NULL)))))))), endpoint1 (object (distingui
shedName (RelativeDistinguishedName (AttributeVal
```

```
ueAssertion (attributeType 1.3.18.0.2.4.6, attrib
uteValue "NETA")), RelativeDistinguishedName (Att
ributeValueAssertion (attributeType 1.3.18.0.0.20
32, attributeValue "SSCP1A")), RelativeDistinguis
hedName (AttributeValueAssertion (attributeType 1
.3.18.0.0.2133, attributeValue "P3A3767A")))), cl
ass 1.3.18.0.0.2085, states 01010100000000, info
(Attribute (attributeId 1.3.18.0.0.2194, attribut
eValue (dependents (and (Dependents (item (distin
guishedName (RelativeDistinguishedName (Attribute
ValueAssertion (attributeType 1.3.18.0.2.4.6, att
ributeValue "NETA")), RelativeDistinguishedName (
AttributeValueAssertion (attributeType 1.3.18.0.0
.2032, attributeValue "SSCP1A")), RelativeDisting
uishedName (AttributeValueAssertion (attributeTyp
e 1.3.18.0.0.2272, attributeValue "NCP.NCP3AB8"))
))), Dependents (item (distinguishedName (Relativ
eDistinguishedName (AttributeValueAssertion (attr
ibuteType 1.3.18.0.2.4.6, attributeValue "NETA"))
, RelativeDistinguishedName (AttributeValueAssert
ion (attributeType 1.3.18.0.0.2032, attributeValu
e "SSCP1A")), RelativeDistinguishedName (Attribut
eValueAssertion (attributeType 1.3.18.0.0.2142, a
ttributeValue "LN3A1"))))))))), Attribute (attrib
uteId 1.3.18.0.0.2142, attributeValue "LN3A1"), A
ttribute (attributeId 1.3.18.0.0.2236, attributeV
alue (noInfo NULL)), Attribute (attributeId 1.3.1
8.0.0.2119, attributeValue (lsAddr C2)), Attribut
e (attributeId 1.3.18.0.0.2121, attributeValue t1
)))))))))
```

The linked-reply in the example, identified by the operation value of 2, contains a set of four instances of the (v1,v2,e1) sequence. Not all instances of the sequence contain all fields of the sequence. The following is a summary of the contents of the four sequences:

First sequence: (v1,v2,e1)

```
vertex 1  :  NETA;SSCP1A (local VTAM host)
                class  : interchangeNode
                states :
                   Operational State    : Enabled
                   Usage State          : Active
                   Administrative State : Unlocked
                   Availability Status  : No Status
                   Procedural Status    : No Status
                   Unknown Status       : False
                   Native Status        : Active
                info :
                   opEquipmentList      : ORGREG.IBM.9021.032082
                   softwareList         : ORGREG.IBM.ACF/VTAM.4.3.0
                   sysplexInfo          : SYSPLEX
                   appnNodeCapabilities : 3340
                   extAppnNodeCap       : 0000
                   subareaLimit         : 511
                   subareaAddress       : 1
                   puName               : ISTPUS
                   gatewaySSCP          : TRUE
                moreInfo :
                   NETA;SSCP1A;0321-L
                     class  : port
                     states :
                        Operational State    : Enabled
                        Usage State          : Active
                        Administrative State : Unlocked
                        Availability Status  : No Status
```

```
                        Procedural Status    : No Status
                        Unknown Status       : False
                        Native Status        : Active
                     info :
                        connectionId       : 0321
                        adapterAddresses   : ()
                        lineType           : nonswitched
                        dlcName            : channel
                        relatedAdapter     : NOINFO NULL
                        dependencies       : NETA;SSCP1A;NCP.ISTPUS,
                                             NETA;SSCP1A

vertex 2  :  NETA;SSCP1A;NCP3AB8 (locally owned NCP)
                class  : t4Node

endpoint 1:  NETA;SSCP1A;1.1.NETA.3.NCP3AB8
                class  : subareaTransmissionGroup

             NETA;SSCP1A;0321-S
                class  : logicalLink
                states :
                   Operational State    : Enabled
                   Usage State          : Active
                   Administrative State : Unlocked
                   Availability Status  : No Status
                   Procedural Status    : No Status
                   Unknown Status       : False
                   Native Status        : Active
                info :
                   portId             : 0321-L
                   partnerConnection  : NETA;NCP3AB8;PU321A
                   adjacentNodeType   : t4   (type 4)
                   dependencies       : NETA;SSCP1A;0321-L
```

This sequence represents the connection from the local VTAM host to a locally owned NCP. This sequence shows:

- The long form of vertex 1 (the attributes for the local VTAM host)
- The long form of the *port* in vertex 1 (channel data)
- The **only** form of vertex 2 (showing the NCP)
- The form of endpoint 1 containing both a *subareaTransmissionGroup* and a *logicalLink*

Second sequence: (v1,e1)

```
vertex 1  :  NETA;SSCP1A;NCP3AB8 (locally owned NCP)
                class  : t4Node
                states :
                   Operational State    : Enabled
                   Usage State          : Active
                   Administrative State : Unlocked
                   Availability Status  : No Status
                   Procedural Status    : No Status
                   Unknown Status       : False
                   Native Status        : Active
                info :
                   subareaLimit       : 255
                   subareaAddress     : 3
                   dependencies       : NETA;SSCP1A;NCP.NCP3AB8
                                        NETA;SSCP1A
                moreInfo :
                   NETA;SSCP1A;LN3A6 (SDLC line in NCP major node)
                     class  : port
                     states :
                        Operational State    : Enabled
                        Usage State          : Active
                        Administrative State : Unlocked
```

```
                        Availability Status  : No Status
                        Procedural Status    : No Status
                        Unknown Status       : False
                        Native Status        : Active
                     info :
                        connectionId       : 0303
                        adapterAddresses   : ()
                        lineType           : nonswitched
                        dlcName            : SDLC
                        relatedAdapter     : NOINFO NULL
                        dependencies       : NETA;SSCP1A;NCP.NCP3AB8,
                                             NETA;SSCP1A;NCP3AB8

endpoint 1:  NETA;SSCP1A;P3A3274E (PU defined under LN3A6)
                class  : logicalLink
                states :
                   Operational State    : Enabled
                   Usage State          : Active
                   Administrative State : Unlocked
                   Availability Status  : No Status
                   Procedural Status    : No Status
                   Unknown Status       : False
                   Native Status        : Active
                info :
                   portId             : LN3A6
                   partnerConnection  : NOINFO NULL
                   adjacentLinkStationAddress  : LSADDR C2
                   adjacentNodeType   : t20  (type 2.0)
                   dependencies       : NETA;SSCP1A;NCP.NCP3AB8,
                                        NETA;SSCP1A;LN3A6
```

This sequence begins the reporting of the NCP resources; the first line defined in the NCP major node is reported, along with the first of two PUs defined under that line. This sequence shows:

- The long form of vertex 1 (the attributes for the NCP)
- The long form of the *port* in vertex 1 (SDLC line data)
- The line / PU are being used for a connection to a type 2.0 node; VTAM does not report type 2.0 nodes in vertex 2
- The form of endpoint 1 containing only a *logicalLink*; there is no TG information because the contacted node is type 2.0

Third sequence: (v1,v2,e1)

```
vertex 1  :  NETA;SSCP1A;NCP3AB8 (locally owned NCP)
                class  : t4Node

vertex 2  :  NETA;P3A4956G
                class  : lenNode

endpoint 1:  NETA;SSCP1A;0.NETA.P3A4956G
                class  : appnTransmissionGroup
                states :
                   Operational State    : Enabled
                   Usage State          : Active
                   Administrative State : Unlocked
                   Availability Status  : No Status
                   Procedural Status    : No Status
                   Unknown Status       : False
                   Native Status        : N/A
                info :
                  cp-cpSessionSupport   : FALSE
                  appnTGcapabilities    : 00

             NETA;SSCP1A;P3A4956G (PU defined under LN3A6)
                class  : logicalLink
```

```
                  states :
                     Operational State    : Enabled
                     Usage State          : Active
                     Administrative State : Unlocked
                     Availability Status  : No Status
                     Procedural Status    : No Status
                     Unknown Status       : False
                     Native Status        : Active
                  info :
                     portId             : LN3A6
                     partnerConnection  : NOINFO NULL
                     adjacentLinkStationAddress  : LSADDR C4
                     adjacentNodeType   : len
                     dependencies       : NETA;SSCP1A;NCP.NCP3AB8,
                                          NETA;SSCP1A;LN3A6
```

This sequence continues the reporting of the NCP resources by reporting the second PU defined under the previously reported line, which is used for a connection to a LEN node. This sequence shows:

- The short form of vertex 1 (long form of NCP already reported)
- The *port* data for LN3A6 was already reported, so the *port* is omitted from vertex 1
- Vertex 2 contains the contacted LEN node.
- The form of endpoint 1 containing an APPN TG and a *logicalLink*. The *port* associated with this PU, although not provided in vertex 1, can be identified by the *portId* attribute of the *logicalLink*.

Fourth sequence: (v1,e1)

```
vertex 1  :  NETA;SSCP1A;NCP3AB8 (locally owned NCP)
                class    : t4Node
                moreInfo :
                   NETA;SSCP1A;LN3A1 (SDLC line in NCP major node)
                     class  : port
                     states :
                        Operational State    : Enabled
                        Usage State          : Active
                        Administrative State : Unlocked
                        Availability Status  : No Status
                        Procedural Status    : No Status
                        Unknown Status       : False
                        Native Status        : Active
                     info :
                        connectionId       : 0305
                        adapterAddresses   : ()
                        lineType           : nonswitched
                        dlcName            : SDLC
                        relatedAdapter     : NOINFO NULL
                        dependencies       : NETA;SSCP1A;NCP.NCP3AB8,
                                             NETA;SSCP1A;NCP3AB8


endpoint 1:  NETA;SSCP1A;P3A3767A (PU defined under LN3A1)
                class  : logicalLink
                states :
                   Operational State    : Enabled
                   Usage State          : Active
                   Administrative State : Unlocked
                   Availability Status  : No Status
                   Procedural Status    : No Status
                   Unknown Status       : False
                   Native Status        : Active
                info :
                   portId             : LN3A1
```

```
partnerConnection  : NOINFO NULL
adjacentLinkStationAddress  : LSADDR C2
adjacentNodeType   : t1   (type 1)
dependencies       : NETA;SSCP1A;NCP.NCP3AB8,
                     NETA;SSCP1A;LN3A1
```

This sequence continues the reporting of the NCP resources by reporting another line, a PU defined under the line, and a connection to a type 1 node. This sequence shows:

- The short form of vertex 1 (long form of NCP already reported)
- The long form of *port* data for LN3A1 is included since it is the initial report of LN3A1.
- The line / PU are being used for a connection to a type 1 node; VTAM does not report connected type 1 nodes in vertex 2.
- The form of endpoint 1 containing only a *logicalLink*. There is no TG information to report for a connection to a type 1 node.

Requesting and monitoring LU data (luCollection)

This section contains the following topics:

Overview

This section describes the action that is used to request monitoring and stop monitoring LU data for a given PU object or agent host object using the *luCollection* managed object class.

Management of LUs requires that a manager application program be able to request the names of all the LUs under a certain PU and to monitor their status. The manager application program can use a *snapshot* action request against the *luCollection* object to get LU information.

The VTAM agent supports two types of *luCollection*

1. *luCollection* against a specified physical unit. This form of *luCollection* is called *luCollection* (PU) and returns all dependent LUs that are defined (either statically or dynamically) under the PU. The physical unit must be defined at the agent host for this command to be successful. For physical units that represent connections to type 2.1 nodes, the independent logical units currently using the PU as an adjacent link station (ALS) for sessions are also reported on the *luCollection snapshot* response.
2. *luCollection* against the agent host. This form of *luCollection* which does not specify a *linkName* in the *luCollection* distinguished name is called *luCollection* (Host) and returns LU resources that are associated with the VTAM agent host. This includes:
   - Application programs
   - CDRSCs
   - USERVARs
   - Generic resources
   - Local non-SNA terminals

The reporting of all CDRSCs for *luCollection* (Host) can generate large amounts of data to be processed because dynamic real CDRSCs as well as predefined CDRSCs can be reported. With the exception of low-entry networking (LEN) independent LUs, these CDRSCs can offer little benefit in terms of managing the network because they represent resources that could be reported by agents at the nodes that own the real resources represented by the CDRSCs. However, these CDRSCs might be of interest in some environments. The user can select which types of CDRSCs are to be included in the *luCollection* (Host) object reported by the VTAM topology agent by specifying one of the following values on the OSITOPO start option:

- ILUCDRSC: Report independent LUs only. This is the default value if not specified.
- ALLCDRSC: Report all CDRSCs, including independent LUs.

Note: This start option does not affect the reporting of independent LUs under *luCollection* (PU).

Action request

A *snapshot* action request is used to request LU data from an agent node. The action is sent as an m-Action-Confirmed operation.

The manager application program can request that any future updates to the *luCollection* object be returned, as they occur. The LU data is requested without updates by specifying the `oneTimeOnly` value in the request. The LU data and any future updates can be requested by specifying the `ongoing` value in the request.

The following example shows a request to the target PU P3A3274A. The following special identifiers are used in the request:

1.3.18.0.0.1811
luCollection
1.3.18.0.0.1815
luCollectionId
1.3.18.0.0.2222
snapshot

```
          msg CMIP-1.ROIVapdu (invokeID 196610, operation-value 7,
argument (baseManagedObjectClass 1.3.18.0.0.1811, baseManagedObjectIns
tance (distinguishedName (RelativeDistinguishedName (AttributeValueAss
ertion (attributeType 1.3.18.0.2.4.6, attributeValue "NETA")), Relativ
eDistinguishedName (AttributeValueAssertion (attributeType 1.3.18.0.0.
2032, attributeValue "SSCP1A")), RelativeDistinguishedName (AttributeV
alueAssertion (attributeType 1.3.18.0.0.2133, attributeValue "P3A3274A
")), RelativeDistinguishedName (AttributeValueAssertion (attributeType
 1.3.18.0.0.1815, attributeValue "luCollection")))), actionInfo (actio
nType 1.3.18.0.0.2222, actionInfoArg (start ongoing)))))
```

Initial data response

If the `oneTimeOnly` *snapshot* action is requested, all the data (initial data only) is returned in action linked replies. To indicate that the data for the entire set of LUs has been returned, the agent sends an additional ROIV action linked reply that is an empty set. Then an RORS action response is sent with only the invoke identifier of the original action request.

If the `ongoing` *snapshot* action was requested, all the initial data is returned in action linked replies, as for the `oneTimeOnly` *snapshot* action. To indicate that the initial data for the entire set of LUs has been returned, the VTAM topology agent sends an additional ROIV action linked reply that is an empty set. The VTAM topology agent is then ready to process updates for the *luCollection* object.

Table 14. Reported resources for luCollection (host) initial data

Resource	Object class	Notes
Non-SNA terminal	LU	Local terminal
Application program	LU	VTAM application program. Model not reported.
CDRSC	CDRSC	Dynamic alias never reported. Others affected by OSITOPO start option. Model not reported.
USERVAR	LU Group	Report USERVAR name and value
Generic resource	LU Group	Report generic and real members

Table 15. Reported resources for luCollection (PU) initial data

Resource	Object class	Notes
Dependent LU	LU	Logical Unit
ILU	CDRSC	ILU reported as CDRSC

Note: ILUs that have multiple sessions through the same PU (ALS) will only be reported once.

Update data response

When the `ongoing` *snapshot* action has been issued and is currently in effect, updates to the *luCollection* object results in the sending of a *snapshot* linked reply with the update. In this case, only data pertinent to the subject LU flows in the *snapshot* linked reply. In general, updates for *luCollection* (Host) are caused by the activation or inactivation of a major node containing LU or CDRSC definitions or a state change of these resources. Creation or deletion of USERVARs and generic resources also cause updates.

Table 16. Resources with reason for luCollection (host) update data

Resource	Reason	Notes
Non-SNA LU or application program	Creation	Major node activation. Model application ignored.
	Deletion	Major node inactivation. Model application ignored.
	State change	Resource changed state. Model application ignored.
CDRSC	Creation	Dynamic alias ignored. Model CDRSC ignored.
	Deletion	Dynamic alias ignored. Model CDRSC ignored.
	State change	Resource changed state. Model CDRSC ignored.
	Attribute value change	CDRSC has a new owning CDRM. Model CDRSC ignored.
LU Group	Creation	USERVAR or generic resource created
	Deletion	USERVAR or generic resource deleted
	State change	Not applicable
	Attribute value change (member added or member deleted)	Not supported for this *snapshot*

Updates for *luCollection* (PU) are generally caused by state changes of dependent LUs defined under a monitored PU. Also, creation or deletion of dynamic dependent LUs as well as connection of independent LUs using the PU as an adjacent link station (ALS) will result in updates.

Table 17. Resources with reason for luCollection (PU) update data

Resource	Reason	Notes
Dependent LU	Creation	Dynamic LU created
	Deletion	Dynamic LU deleted
	State change	Dependent LU changed state
Independent LU	Creation	New session through adjacent link station
	Deletion	End last session through adjacent link station
	State change	Not reported

Updates for the *luCollection* object can be merged with related updates by the VTAM topology agent before being written to the *snapshot* linked replies.

A *snapshot luUICollection* is automatically cancelled when the PU supporting the *luCollection* is deleted. *luCollection* updates are not merged when they are the result of a VTAM-cancelled *luCollection*.

Some updates for *luCollection snapshot*s might be reported under more than one *luCollection*. For example, some updates, such as updates for independent LUs, might be reported under several monitored PUs and also under the VTAM host.

All VTAM host *luCollection* updates are merged, but updates for a specific PU are not merged if the updates represent independent LUs.

Updates that report a deleted object are not merged.

Action termination

The VTAM topology agent terminates an `ongoing` *snapshot* action for the *luCollection* object under the following conditions:

- A stop *snapshot* action request is received.
- An error occurs during *snapshot* processing in VTAM.
- The association that the *snapshot* is using terminates.
- The target PU object associated with the *luCollection* object is deleted.

When a stop *snapshot* action request is received, the agent sends an ROIV action linked reply for the *snapshot* start request that is an empty set, an RORS action response to the *snapshot* start request, and an RORS action response to the *snapshot* stop request.

luCollection snapshot data

The linked-replies for *luCollection* are made up of multiple instances of the following structure:

```
vertex1
       object  --LU-related object distinguished name
       class   --object class
       states  --OSI states for this object
       info
            dependencies              --object attribute
            residentNodePointer       --object attribute
            nlrResidentNodePointer    --object attribute
            luGroupMembers            --object attribute
            cdrscRealLuName           --object attribute
            userLabel                 --object attribute
       reason                    --reason for this vertex1 to be reported
```

The following list explains what the fields contain.

vertex1
Contains all data reported for a single LU-related object for either initial data or for a single update for the object.

object Distinguished name of the LU-related object.

class Monitored LU-related objects are reported under the following object classes:
- *logicalUnit*
- *crossDomainResource*
- *luGroup* (For USERVAR or generic resource)

states 14-character string for one of the following OSI states:
- *operationalState*
- *usageState*
- *administrativeState*
- *availabilityStatus*
- *proceduralStatus*
- *unknownStatus*
- *nativeStatus*

No states information is returned for *luGroup*

info Set of attributes for the LU-related object. Not all attributes are reported for

all object classes or are necessarily reported in the order shown in the following tables Table 18 and Table 19. The following table shows which attributes might be reported for an object class.

Table 18. Attributes for luCollection (host) reported objects

Attribute	LU	CDRSC	luGroup
dependencies	X	X	
residentNodePointer	X		
nlrResidentNodePointer		X	
luGroupMembers			X
cdrscRealLuName		X	
userLabel	X	X	
tn3270DnsName	X		
tn3270IpAddress	X		
tn3270portNumber	X		

Table 19. Attributes for luCollection (PU) reported objects

Attribute	LU	CDRSC
dependencies	X	X
residentNodePointer	X	
nlrResidentNodePointer		X
cdrscRealLuName		X

reason Indicates why the *snapshot* update is being sent:

Note: The `reason` field is omitted when the intended value is `addOrUpdate`.

Value Description

deleted
Object is deleted.

addOrUpdate
Object is added or changed. The default is addOrUpdate.

luCollection (PU) snapshot example

The following example shows an initial data response for the target PU P3A3274A. This example shows only vertex1 data for LU L3A3278A:

```
                                                msg CMIP-1.ROIVapdu
(invokeID 131075, linked-ID 196610, operation-value 2, argument (actio
nResult (managedObjectClass 1.3.18.0.0.1811, managedObjectInstance (di
stinguishedName (RelativeDistinguishedName (AttributeValueAssertion (a
ttributeType 1.3.18.0.2.4.6, attributeValue "NETA")), RelativeDistingu
ishedName (AttributeValueAssertion (attributeType 1.3.18.0.0.2032, att
ributeValue "SSCP1A")), RelativeDistinguishedName (AttributeValueAsser
tion (attributeType 1.3.18.0.0.2133, attributeValue "P3A3274A")), Rela
tiveDistinguishedName (AttributeValueAssertion (attributeType 1.3.18.0
.0.1815, attributeValue "luCollection")))), actionReply (actionType 1.
3.18.0.0.2222, actionReplyInfo ((vertex1 (object (distinguishedName (R
elativeDistinguishedName (AttributeValueAssertion (attributeType 1.3.1
8.0.2.4.6, attributeValue "NETA")), RelativeDistinguishedName (Attribu
teValueAssertion (attributeType 1.3.18.0.0.2032, attributeValue "SSCP1
A")), RelativeDistinguishedName (AttributeValueAssertion (attributeTyp
e 1.3.18.0.0.1984, attributeValue "NETA.L3A3278A")))), class 1.3.18.0.
```

```
0.1829, states 01000100000000, info (Attribute (attributeId 1.3.18.0.0
.2194, attributeValue (dependents (and (Dependents (item (distinguishe
dName (RelativeDistinguishedName (AttributeValueAssertion (attributeTy
pe 1.3.18.0.2.4.6, attributeValue "NETA")), RelativeDistinguishedName
(AttributeValueAssertion (attributeType 1.3.18.0.0.2032, attributeValu
e "SSCP1A")), RelativeDistinguishedName (AttributeValueAssertion (attr
ibuteType 1.3.18.0.0.2272, attributeValue "NCP.NCP3AB8"))))), Dependen
ts (item (distinguishedName (RelativeDistinguishedName (AttributeValue
Assertion (attributeType 1.3.18.0.2.4.6, attributeValue "NETA")), Rela
tiveDistinguishedName (AttributeValueAssertion (attributeType 1.3.18.0
.0.2032, attributeValue "SSCP1A")), RelativeDistinguishedName (Attribu
teValueAssertion (attributeType 1.3.18.0.0.2133, attributeValue "NETA.
P3A3274A"))))))))), Attribute (attributeId 1.3.18.0.0.2018, attributeV
alue (distinguishedName (RelativeDistinguishedName (AttributeValueAsse
rtion (attributeType 1.3.18.0.2.4.6, attributeValue "NETA")), Relative
DistinguishedName (AttributeValueAssertion (attributeType 1.3.18.0.0.2
032, attributeValue "SSCP1A")), RelativeDistinguishedName (AttributeVa
lueAssertion (attributeType 1.3.18.0.0.2133, attributeValue "NETA.P3A3
274A"))))))))))))
```

The following translated initial data shows LU L3A3278A is active but not used, under PU P3A3274A. The *userLabel* attribute is not reported because the LU is not an application with an ACBNAME.

```
luCollection object name: NETA;SSCP1A;P3A3274A;luCollection

vertex 1: NETA;SSCP1A;NETA.L3A3278A

          class             : logicalUnit
          states
             Operational State   : Enabled
             Usage State         : Idle
             Administrative State: Unlocked
             Availability Status : No Status
             Procedural Status   : No Status
             Unknown Status      : False
             Native Status       : Active

          info
             residentNodePointer      : NETA;SSCP1A;NETA.P3A3274A
             dependencies:            : NETA;SSCP1A;NCP.NCP3AB8
                                        NETA;SSCP1A;NETA.P3A3274A
```

The following example shows update data for the target PU P3A3274A. The example includes vertex1 data for LU L3A3278A. The following special identifiers are used in the responses:

1.3.18.0.0.1811
luCollection

1.3.18.0.0.1815
luCollectionId

1.3.18.0.0.2222
snapshot

```
                                                    msg CMIP-1.ROIVapdu
(invokeID 131077, linked-ID 196610, operation-value 2, argument (actio
nResult (managedObjectClass 1.3.18.0.0.1811, managedObjectInstance (di
stinguishedName (RelativeDistinguishedName (AttributeValueAssertion (a
ttributeType 1.3.18.0.2.4.6, attributeValue "NETA")), RelativeDistingu
ishedName (AttributeValueAssertion (attributeType 1.3.18.0.0.2032, att
ributeValue "SSCP1A")), RelativeDistinguishedName (AttributeValueAsser
tion (attributeType 1.3.18.0.0.2133, attributeValue "P3A3274A")), Rela
tiveDistinguishedName (AttributeValueAssertion (attributeType 1.3.18.0
.0.1815, attributeValue "luCollection")))), actionReply (actionType 1.
3.18.0.0.2222, actionReplyInfo ((vertex1 (object (distinguishedName (R
elativeDistinguishedName (AttributeValueAssertion (attributeType 1.3.1
8.0.2.4.6, attributeValue "NETA")), RelativeDistinguishedName (Attribu
```

```
teValueAssertion (attributeType 1.3.18.0.0.2032, attributeValue "SSCP1
A")), RelativeDistinguishedName (AttributeValueAssertion (attributeTyp
e 1.3.18.0.0.1984, attributeValue "NETA.L3A3278A")))), class 1.3.18.0.
0.1829, states 01010100000001, info ()))))))))
```

The following translated update data shows a session with LU L3A3278A was started. No attributes are reported since this update represents a state change.

```
luCollection object name: NETA;SSCP1A;P3A3274A;luCollection

vertex 1: NETA;SSCP1A;NETA.L3A3278A

          class             : logicalUnit
          states
            Operational State        : Enabled
            Usage State              : Active
            Administrative State     : Unlocked
            Availability Status      : No Status
            Procedural Status        : No Status
            Unknown Status           : False
            Native Status            : Active with session
```

The following example shows attribute data for APPL APPL1A which represents a TN3270 connection with an IP address, DNS and port.

```
                        msg CMIP-1.RORSapdu (invokeID 196616, re
sultOption (operation-value 3, result (managedObjectClass 1.
3.18.0.0.1829, managedObjectInstance (distinguishedName (Rel
ativeDistinguishedName (AttributeValueAssertion (attributeTy
pe 1.3.18.0.2.4.6, attributeValue "NETA")), RelativeDistingu
ishedName (AttributeValueAssertion (attributeType 1.3.18.0.0
.2032, attributeValue "SSCP1A")), RelativeDistinguishedName
(AttributeValueAssertion (attributeType 1.3.18.0.0.1984, att
ributeValue "NETA.APPL1")))), currentTime "2002/11/07-08:47:
22.0", attributeList (Attribute (attributeId 2.9.3.2.7.31, a
ttributeValue unlocked), Attribute (attributeId 2.9.3.2.7.50
, attributeValue ()), Attribute (attributeId 1.2.124.360501.
1.209, attributeValue ()), Attribute (attributeId 2.9.3.2.7.
33, attributeValue ()), Attribute (attributeId 1.3.18.0.0.21
94, attributeValue (dependents (and (Dependents (item (disti
nguishedName (RelativeDistinguishedName (AttributeValueAsser
tion (attributeType 1.3.18.0.2.4.6, attributeValue "NETA")),
 RelativeDistinguishedName (AttributeValueAssertion (attribu
teType 1.3.18.0.0.2032, attributeValue "SSCP1A")), RelativeD
istinguishedName (AttributeValueAssertion (attributeType 1.3
.18.0.0.2272, attributeValue "APPL.APPL1A"))))), Dependents
(item (distinguishedName (RelativeDistinguishedName (Attribu
teValueAssertion (attributeType 1.3.18.0.2.4.6, attributeVal
ue "NETA")), RelativeDistinguishedName (AttributeValueAssert
ion (attributeType 1.3.18.0.0.2032, attributeValue "SSCP1A")
)))))))), Attribute (attributeId 1.2.124.360501.1.240, attri
buteValue (string "APPL1")), Attribute (attributeId 1.3.18.0
.0.1984, attributeValue "NETA.APPL1"), Attribute (attributeI
d 1.3.18.0.0.1819, attributeValue "APPL1"), Attribute (attri
buteId 2.9.3.2.7.63, attributeValue 1.3.18.0.0.1911), Attrib
ute (attributeId 1.3.18.0.0.2080, attributeValue 0), Attribu
te (attributeId 2.9.3.2.7.65, attributeValue 1.3.18.0.0.1829
), Attribute (attributeId 2.9.3.2.7.35, attributeValue enabl
ed), Attribute (attributeId 2.9.3.2.7.66, attributeValue (OB
JECT-IDENTIFIER 2.9.3.2.4.16, OBJECT-IDENTIFIER 2.9.3.2.4.17
, OBJECT-IDENTIFIER 0.0.13.3100.0.4.1, OBJECT-IDENTIFIER 1.3
.14.2.2.2.40, OBJECT-IDENTIFIER 1.3.14.2.2.2.35, OBJECT-IDEN
TIFIER 1.3.18.0.0.1871, OBJECT-IDENTIFIER 1.3.18.0.0.2066, O
BJECT-IDENTIFIER 1.3.18.0.0.1818, OBJECT-IDENTIFIER 1.2.124.
360501.10.62, OBJECT-IDENTIFIER 1.3.18.0.0.7898)), Attribute
 (attributeId 2.9.3.2.7.36, attributeValue ()), Attribute (a
ttributeId 1.3.18.0.0.2018, attributeValue (distinguishedNam
```

```
e (RelativeDistinguishedName (AttributeValueAssertion (attri
buteType 1.3.18.0.2.4.6, attributeValue "NETA")), RelativeDi
stinguishedName (AttributeValueAssertion (attributeType 1.3.
18.0.0.2032, attributeValue "SSCP1A")))))), Attribute (attrib
uteId 1.2.124.360501.1.302, attributeValue (noResources none
)), Attribute (attributeId 1.3.18.0.0.7901, attributeValue (
ipv6 "8:7:6:5:4:3:2:1%ZONE1ABC")), Attribute (attributeId 1.
3.18.0.0.7902, attributeValue (portNumber "1027")), Attribut
e (attributeId 1.3.18.0.0.7900, attributeValue (fullName "AB
CDEFGH")), Attribute (attributeId 2.9.3.2.7.38, attributeVal
ue FALSE), Attribute (attributeId 2.9.3.2.7.39, attributeVal
ue idle), Attribute (attributeId 0.0.13.3100.0.7.50, attribu
teValue "APPL1")))))
```

The following translated data shows an IPv6 type IP address with zone, the DNS and port.

```
Name              : NETA.SSCP1A.NETA.APPL1(luName)
  Class             : LU
  Attribute List :

  administrativeState           : UNLOCKED
  attachedCircuitList           : ()
  availabilityStatus            : ()
  functionId                    : APPL1
  luName                        : NETA.APPL1
  luSecondName                  : APPL1
  nameBinding                   : LU_not_TYPE5
  nativeStatus                  : ACT
  objectClass                   : LU
  operationalState              : ENABLED
  proceduralStatus              : ()
  residentNodePointer           : NETA.SSCP1A(snaNodeName)
  tn3270ClientIpAddress         : 8:7:6:5:4:3:2:1%ZONE1ABC
  tn3270ClientPortNumber        : 1027
  tn3270ClientDnsName           : ABCDEFGH
  unknownStatus                 : FALSE
  usageState                    : IDLE
  userLabel                     : APPL1
```

Monitoring resources through event reports

This section contains the following topics:

Overview

A manager application program can monitor certain resource objects maintained by the VTAM topology agent for certain defined events. The VTAM topology agent uses the event reporting in CMIP services to monitor resources.

The resource monitoring process consists of:

- Management of the event reporting environment
- Notification of events from the agent application program to the manager application program

Management of the event reporting environment

The event reporting environment is mostly determined by the set of **event forwarding discriminator (EFD)** objects that exist at any given time. The VTAM topology agent does not become involved with the management of the event reporting environment. CMIP services handles the creation and maintenance of the event reporting environment.

The event reporting environment for the VTAM topology agent is determined partly by the OSIEVENT start option. For a description of the factors that control the event reporting environment, refer to "Special considerations for topology manager application programs" on page 14.

Creation of the event forwarding discriminator

The manager application program can create an EFD object at CMIP services by sending a CMIP create request specifying:

- A managed object class (EFD)
- A managed object instance
- The discriminator construct that is used to filter notifications
- The destination that contains the name of the object (application program) that should receive event reports

The following example shows a CMIP create EFD request; the discriminator construct passes only LU group change notifications.

The following special identifiers are used in the request:

2.9.3.2.3.4
eventForwardingDiscriminator

2.9.3.2.7.1
discriminatorId

2.9.3.2.7.56
discriminatorConstruct

2.9.3.2.7.14
eventType

1.3.18.0.0.1810
luGroupChangeNotif

2.9.3.2.7.55
destination

1.3.18.0.0.2175
managerApplicationName

```
      msg CMIP-1.ROIVapdu (invokeID 196611, operation-value 8,
(managedObjectClass 2.9.3.2.3.4,(managedObjectInstance (distinguishedN
ame (RelativeDistinguishedName (AttributeValueAssertion (attributeType
 1.3.18.0.2.4.6, attributeValue NETA)), RelativeDistinguishedName (Att
ributeValueAssertion (attributeType 2.9.3.2.7.4, attributeValue (name
SSCP1A))), RelativeDistinguishedName (AttributeValueAssertion (attribu
teType 2.9.3.2.7.1, attributeValue (string luGrCh))))),attributeList(
(attributeId 2.9.3.2.7.31,attributeValue unlocked),(attributeId 2.9.3.
2.7.56,attributeValue (item (equality (attributeId 2.9.3.2.7.14,attrib
uteValue 1.3.18.0.0.1810)))),(attributeId 2.9.3.2.7.55,attributeValue
(single (name (RDNSequence (RelativeDistinguishedName (AttributeValueA
ssertion (attributeType 1.3.18.0.2.4.6, attributeValue NETA)), Relativ
eDistinguishedName (AttributeValueAssertion (attributeType 2.9.3.2.7.4
, attributeValue (name SSCP1A))), RelativeDistinguishedName (Attribute
ValueAssertion (attributeType 1.3.18.0.0.2175, attributeValue Manager)
)))))))))
```

Reporting events to the manager application program

Once the resource monitoring environment is created, unsolicited management information can flow from the agent to the manager. When certain defined events occur, a resource object sends a notification. The VTAM topology agent sends this notification data to CMIP services. CMIP services applies the filtering constructs based on active EFDs to the notification message. If it is determined that the event matches a filter, CMIP services determines which manager application program should receive the message in an event report. Note that multiple event reports might be sent to multiple destinations based on a single notification event. This situation might occur when multiple EFD filters are satisfied by a particular notification event.

The event notification data is subject to the same merge process used for the *snapshot* data. See "ACTION(snapshot) update merging" on page 168 for a description of the merge process. The merging of event data is based on the state of the resource being reported. If the state to be reported is a transient state (non-resting state), the event data is held and merged with other event data until the resource is reporting event data with a resting state. As with the *snapshot* merge process, the event merging is timed such that resources that remain in transient states too long are reported.

Merging event data has an additional consideration that does not apply to the *snapshot* update data: the event data specifies the reason for the event. It is possible for event data with different reasons to be merged. In that case, the general rule used by the VTAM topology agent is that the reason for event data that was merged last is used in the reported notification. For example, if an object-creation event occurred but the state was a transient state, the event data is held. Subsequent state-change event data might be merged with the object-creation data. When the event notification is finally sent, the reason specifies state-change. The manager application program can infer the creation of the object by having no previous report of the object.

CMIP services sends the notification data to the manager in the form of a CMIP event-report request. VTAM supports unconfirmed event-reports only (m-EventReport).

Changes affecting network resources result in notifications that might flow to a manager application program. The VTAM topology agent supports notifications for:

- State change
- Object creation
- Object deletion
- LU group change

Event report data

Data in the m-EventReport request appear in the following structure:

```
        managedObjectClass          --object class
        managedObjectInstance       --object distinguished name
        eventTime                   --time stamp
        eventType                   --event type

  --No further info for objectCreation and objectDeletion

  --Added for stateChange eventType
        eventInfo
               attributeIdentifierList            --new attribute identifiers
```

```
                stateChangeDefinition
                          nativeStatus              --object attribute
                          operationalState          --object attribute
                          usageState                --object attribute
                          availabilityStatus        --object attribute
                          proceduralStatus          --object attribute
                          unknownStatus             --object attribute

--Added for luGroupChangeNotif eventType
         eventInfo
                notifReason          --reason for LU group change
                luName               --group member name
                luGroupSize          --group size
```

managedObjectClass
Object classes for the reported resource.

managedObjectInstance
Distinguished name of the reported resource.

eventTime
Time stamp with the current system time when the event report is built from the corresponding notification.

eventType
One of the following events:

- *stateChange*
- *objectCreation*
- *objectDeletion*
- *luGroupChangeNotif* (for USERVARs, Generic Resources, and IP info attributes for TN3270 connection LUs.)

  ObjectCreation of an luGroup object is considered an *luGroupChangeNotif* event. The VTAM topology agent reports the member that caused the object to be created. The manager application program can infer the creation of the object by monitoring for this event report.

attributeIdentifierList
List of attributes identifiers, also provided in the following stateChangeDefinition field, for which new values are provided.

stateChangeDefinition
Contains six out of seven OSI states in the attribute form, rather than in the OCTET string form. The VTAM topology agent returns all attributes with the current values.

administrativeState is omitted because the value is always assumed to be `unlocked`.

notifReason
One of the following reasons for LU group change.

- luAdded
- luDeleted

luName
Name of member in LU group that caused the event.

luGroupSize
Number of members in LU group.

Event report example

The following example shows an event report for an LU group change.

The following special identifiers are used in the event report:

1.3.18.0.0.1803
luGroup

1.3.18.0.0.1807
luGroupName

1.3.18.0.0.1810
luGroupChangeNotif

```
        msg CMIP-1.ROIVapdu (invokeID 65541, operation-value 0,
 argument (managedObjectClass 1.3.18.0.0.1803, managedObjectInstance (d
 istinguishedName (RelativeDistinguishedName (AttributeValueAssertion (
 attributeType 1.3.18.0.2.4.6, attributeValue "NETA")), RelativeDisting
 uishedName (AttributeValueAssertion (attributeType 1.3.18.0.0.2032, at
 tributeValue "SSCP1A")), RelativeDistinguishedName (AttributeValueAsse
 rtion (attributeType 1.3.18.0.0.1807, attributeValue "GENERIC1")))), e
 ventTime "1994/10/17-11:08:18.0", eventType 1.3.18.0.0.1810, eventInfo
  (notifReason luAdded, luName "NETA.APPL2", luGroupSize 1)))
```

The following translated event data shows that LU NETA.APPL2 has been added to the LU group GENERIC1 (a generic resource name).

```
Resource name     : NETA;SSCP1A;GENERIC1
class             : luGroup
eventTime         : 1994/10/17-11:08:18.0
eventType         : luGroupChangeNotif
eventInfo
     notifReason     : luAdded
     luName          : NETA.APPL2
     luGroupSize     : 1
```

The following example shows an event report for a state change.

The following special identifiers are used in the event report:

1.3.18.0.0.1829
logicalUnit

2.9.3.2.10.14
stateChange

```
        msg CMIP-1.ROIVapdu (invokeID 65551, operation-value 0,
 argument (managedObjectClass 1.3.18.0.0.1829, managedObjectInstance (d
 istinguishedName (RelativeDistinguishedName (AttributeValueAssertion (
 attributeType 1.3.18.0.2.4.6, attributeValue "NETA")), RelativeDisting
 uishedName (AttributeValueAssertion (attributeType 1.3.18.0.0.2032, at
 tributeValue "SSCP1A")), RelativeDistinguishedName (AttributeValueAsse
 rtion (attributeType 1.3.18.0.0.1984, attributeValue "NETA.APPL2")))),
  eventTime "1995/01/27-14:16:20.0", eventType 2.9.3.2.10.14, eventInfo
  (attributeIdentifierList (AttributeId 1.3.18.0.0.2080, AttributeId 2.
 9.3.2.7.35, AttributeId 2.9.3.2.7.39, AttributeId 2.9.3.2.7.33, Attrib
 uteId 2.9.3.2.7.36, AttributeId 2.9.3.2.7.38), stateChangeDefinition (
 (attributeID 1.3.18.0.0.2080, newAttributeValue 0), (attributeID 2.9.3
 .2.7.35, newAttributeValue enabled), (attributeID 2.9.3.2.7.39, newAtt
 ributeValue idle), (attributeID 2.9.3.2.7.33, newAttributeValue ()), (
 attributeID 2.9.3.2.7.36, newAttributeValue ()), (attributeID 2.9.3.2.
 7.38, newAttributeValue FALSE))))))
```

The following translated event data shows that LU NETA.APPL2 has changed state. This event report example was reported because of an EFD set to monitor state changes.

```
Resource name     : NETA;SSCP1A;NETA.APPL2
Class             : logicalUnit
eventTime         : 1995/01/27-14:16:20.0
eventType         : stateChange
AttributeIdentifierList
   nativeStatus
   operationalState
   usageState
   availabilityStatus
   proceduralStatus
   unknownStatus
StateChangeDefinition
   nativeStatus             : newAttributeValue ACTIVE
   operationalState         : newAttributeValue ENABLED
   usageState               : newAttributeValue IDLE
   availabilityStatus       : newAttributeValue ()
   proceduralStatus         : newAttributeValue ()
   unknownStatus            : newAttributeValue FALSE
```

Chapter 16. Requesting specific resource data

This chapter describes how the VTAM topology agent gathers information about specific resources. The following topics are included:
- Requesting specific resource data (GET)
- Requesting specific resource data (logicalUnitIndex)

Requesting specific resource data (GET)

This section contains the following topics:

Overview

The CMIP GET operation can be used by a manager application program to obtain resource information for a single named resource. The object class of the resource might not be known, so it is not necessary that the manager application program know the exact object class of a particular resource before using the GET operation to obtain information about that resource. It is necessary, however, that the manager application program know the name of the resource and that the name be properly constructed for the object class containing the resource.

If the VTAM topology agent receives a GET request for a known resource, the VTAM topology agent provides a single GET response containing the requested resource information, as it exists when the GET request is processed.

GET request

The GET request contains the following information:

object class

If the object class of the resource is known, it can be specified in the request as the object identifier (OI) representing the appropriate class. If the class is not known, the class can be specified as *actualClass* (2.9.3.4.3.42). *actualClass* is a special class specification that tells the VTAM topology agent that the class is unknown and that the VTAM topology agent should return the real object class in the GET response.

The object class can be specified as a class higher in the inheritance tree than the real class of the resource. An object in the lower (real) class can validly respond to a request as if the object were in the higher (requested) class. Another way to state this behavior is that the lower class can act allomorphically to the higher class; it can emulate the higher class. For example, if a VTAM node is an interchange node (combination type 5 node and APPN network node) and the VTAM topology agent at that VTAM node receives a GET request for the VTAM node that specifies the *t5Node* object class, the VTAM topology agent at that node can respond to the request since the *interchangeNode* class inherits from the *t5Node* class, allowing an object in the *interchangeNode* class to act allomorphically to the *t5Node* class. If the VTAM node is a *t5Node* and the VTAM topology agent

received a request specifying an *interchangeNode*, the request is rejected, since a *t5Node* cannot act as an *interchangeNode*.

object instance
The distinguished name of the object must be provided in the GET request. Although the object class might not be known, the naming attribute used in the object instance name must be a valid naming attribute for the object class in which the resource exists. No generic form exists for naming attribute (similar to *actualClass*) that can apply to any class. Note that the first two relative distinguished names (RDNs) in the distinguished name must be the netID and node name of the VTAM topology agent host.

scope The VTAM topology agent does not support the scoping function; therefore, the optional scope information can be omitted from the GET request. If scope is specified in the request, the associated scope value must be specified as '(basicScope 0)'.

filter The VTAM topology agent does not support the filtering function; therefore, the optional filter information can be omitted from the GET request. If the filter is specified in the request, the associated filter value must be specified as either '(and ())' or '(or ())'.

attribute list
The attribute list contains a set of OIs representing the attributes of the object for which the manager application program is requesting information. The attributes must be defined in the requested object class, or the discovered object class if *actualClass* is specified. They must also be among the attributes supported by the VTAM topology agent. For the list of supported attributes, refer to Appendix E, "VTAM topology agent object and attribute tables," on page 301. The attribute list can be omitted from the GET request. Omitting it indicates to the VTAM topology agent that *all* supported attributes for the object class (specified or discovered) must be provided in the GET response.

The VTAM topology agent supports GET requests for resources in the following object classes:
- *appnEN*
- *appnNN*
- *appnRegisteredLu*
- *crossDomainResource*
- *definitionGroup*
- *interchangeNode*
- *lenNode*
- *logicalLink*
- *logicalUnit*
- *luGroup*
- *migrationDataHost*
- *port*
- *t2-1Node*
- *t4Node*
- *t5Node*

Certain object classes are named with netID and snaNodeName; the object instance names for objects in these classes must be the VTAM topology agent host name for GET requests for these objects to be routed to the VTAM topology agent. For example, a VTAM topology agent at node NETA;SSCP1A might report node NETA;SSCP2A as a type 5 node in a *snaLocalTopo* response. A subsequent GET request sent to object NETA;SSCP2A will not be routed to the VTAM topology

agent at node NETA;SSCP1A because the object instance name in the GET request is not named under NETA;SSCP1A, the VTAM topology agent name. This situation applies to the following object classes:

- *appnEN*
- *appnNN*
- *interchangeNode*
- *migrationDataHost*
- *t5Node*

Network-qualified names and GET requests

GET requests for *logicalLink*, *logicalUnit*, *crossDomainResource*, and *appnRegisteredLU* can specify a network-qualified name as the third RDN in the distinguished name. The name can also be specified without being network qualified. The VTAM topology agent determines whether the name matches the name of a resource at the VTAM topology agent host and verifies that the resource found is the correct type for the naming attribute specified in the GET request.

Rules for matching names by object type are listed here:

logicalLink
: If the *linkName* attribute is network qualified, the netID must be the same as the netID of the VTAM topology agent host receiving the GET request.

logicalUnit
: If the *luName* attribute is network qualified, the netID must be the same as the netID that VTAM display commands require for displaying the resource. The VTAM topology agent host netID will always work. For dependent LUs that are attached to a non-native PU type 2.1 node (nonnative network attachment), the netID of the physical unit will also be accepted. A non-network qualified name always has a default netID that is the same as the VTAM topology agent host netID.

  For application logical units, the *luName* attribute can specify the name coded on the ACBNAME operand of the APPL definition statement. For this name to match, it must be either non-network-qualified or network qualified with the netID of the VTAM topology agent host. When a match occurs, the VTAM topology agent host returns data for the application program with the specified ACBNAME.

crossDomainResource
: If the *nonLocalResourceName* attribute is specified with a network-qualified name, the name and netID must exactly equal the name and netID of the CDRSC for a match to occur. If a non-network-qualified name is given, the name will only match a CDRSC that has been added to the VTAM topology agent host SRT directory.

  The *nonLocalResourceName* attribute can also specify the name coded on the LUALIAS operand of a CDRSC definition. For this name to match, the name must be either non-network-qualified or qualified with the netID of the VTAM topology agent host.

appnRegisteredLU
: If the *nonLocalResourceName* attribute is specified with a network-qualified name, the name and netID must exactly match the name and netID of the registered LU. If a non-network-qualified name is given, the name will only match a registered LU with the netID of the VTAM topology agent host.

The following example shows a GET request for resource data:

```
msg CMIP-1.ROIVapdu (invokeI D 196610, operation-
value 3, argument (baseManagedObjectClass 1.3.18.
0.0.1844, baseManagedObjectInstance (distinguishe
dName (RelativeDistinguishedName (AttributeValueA
ssertion (attributeType 1.3.18.0.2.4.6, attribute
Value "NETA")), RelativeDistinguishedName (Attrib
uteValueAssertion (attributeType 1.3.18.0.0.2032,
 attributeValue "SSCP1A")), RelativeDistinguished
Name (AttributeValueAssertion (attributeType 1.3.
18.0.0.2032, attributeValue "NCP3AB8")))), attrib
uteIdList (AttributeId 2.9.3.2.7.33, AttributeId
1.3.18.0.0.2035, AttributeId 1.3.18.0.0.2036, Att
ributeId 1.3.18.0.0.1971, AttributeId 1.3.18.0.0.
2080, AttributeId 2.9.3.2.7.35, AttributeId 2.9.3
.2.7.36, AttributeId 2.9.3.2.7.39)))
```

Note from the example that the class specified is *t4Node* and that the object instance is NETA;SSCP1A;NCP3AB8. There is no scope or filter specified. The attribute list is specified and contains the following attributes:

- *availabilityStatus*
- *subareaAddress*
- *subareaLimit*
- *gatewayNode*
- *nativeStatus*
- *operationalState*
- *proceduralStatus*
- *usageState*

GET response

When the VTAM topology agent successfully processes a GET request that contains only valid data, the GET response is in the form of a single RORS message. The RORS specifies the object class, object instance, and the list of requested attributes and their values, as they exist when the GET request is processed.

If the real object class is found to be a valid allomorph for the requested object class, then the list of attributes reported in the response is limited to attributes defined in the **requested class**. For example, if requested class is *t5Node* and an attribute list is not specified in the request, and if the resource is discovered to be an *interchangeNode*, the GET response includes an object class of *interchangeNode*, the same object instance name as the request, and the list of attributes defined in the *t5Node* object class. Attributes of the *interchangeNode* that are not defined also in the *t5Node* class are not provided in the response.

If the requested resource is found to be in an object class other than the class specified, and the discovered class is **not** a valid allomorph for the requested class, then the GET response is an ROER error message, with error value of `noSuchObjectInstance`.

If the naming attribute specified in the object instance name is not valid for the object class specified, the GET response is an ROER, with error value of `noSuchObjectInstance`. If *actualClass* is specified in the request, and the naming attribute specified in the object instance is not valid for the discovered object class, then the GET response is an ROER error message, with error value of `noSuchObjectInstance`.

If an attribute list is specified in the GET request, and one of more of the specified attributes are not defined in the requested or discovered class, then the GET

response is an ROER, with error value indicating a `getListError` error. The `getListError` syntax specifies that each requested attribute be listed in the response, with an indication of whether the attribute is valid or not valid. The valid atttributes are then accompanied by their requested values.

If the resource name specified in the object instance of a request is not known to VTAM, the GET response is an ROER message, with error value indicating a `noSuchObjectInstance`.

Note that it is possible to specify the same attribute multiple times in the attribute list in the GET request. If valid attributes are specified multiple times, the VTAM topology agent ignores the secondary specifications and returns each attribute value only one time in the GET response. If, however, attributes that are not valid are specified multiple times in the GET request, the VTAM topology agent indicates an attribute error in the GET response for each attribute that is not valid.

GET data

The following shows the major data fields that comprise the GET response:

- object class
- object instance
- attribute list:
  - attribute ID
  - attribute value

GET data example

The following is an example of the GET response that the VTAM topology agent may return for the GET request example shown previously.

```
                    msg CMIP-1.RORSapdu (invokeID
 196610, resultOption (operation-value 3, result
(managedObjectClass 1.3.18.0.0.1844, managedObjec
tInstance (distinguishedName (RelativeDistinguish
edName (AttributeValueAssertion (attributeType 1.
3.18.0.2.4.6, attributeValue "NETA")), RelativeDi
stinguishedName (AttributeValueAssertion (attribu
teType 1.3.18.0.0.2032, attributeValue "SSCP1A"))
, RelativeDistinguishedName (AttributeValueAssert
ion (attributeType 1.3.18.0.0.2032, attributeValu
e "NCP3AB8")))), currentTime "1995/04/24-10:03:50
.0", attributeList (Attribute (attributeId 2.9.3.
2.7.33, attributeValue ()), Attribute (attributeI
d 1.3.18.0.0.1971, attributeValue FALSE), Attribu
te (attributeId 1.3.18.0.0.2080, attributeValue 0
), Attribute (attributeId 2.9.3.2.7.35, attribute
Value enabled), Attribute (attributeId 2.9.3.2.7.
36, attributeValue ()), Attribute (attributeId 1.
3.18.0.0.2035, attributeValue 3), Attribute (attr
ibuteId 1.3.18.0.0.2036, attributeValue 255), Att
ribute (attributeId 2.9.3.2.7.39, attributeValue
active)))))
```

Note that the requested attributes were all returned with the following values:

```
availabilityStatus  :  ()
gatewayNode         :  FALSE
nativeStatus        :  0   (means active)
operationalState    :  enabled
proceduralStatus    :  ()
subareaAddress      :  3
subareaLimit        :  255
usageState          :  active
```

Requesting specific resource data (logicalUnitIndex)

This section contains the following topics:

Overview

The *logicalUnitIndex* collection object allows a user to request a *snapshot* for all LU-related resources that match a given name.

logicalUnitIndex is used to report on LUs, CDRSCs, USERVARs and generic resources, without forcing a manager application program to understand the actual resource class in advance. When a *snapshot* is issued against the *logicalUnitIndex* object, certain information on all resources that match the name supplied by the *snapshot* and that are LUs, CDRSCs, USERVARs or generic resources is returned.

For *logicalUnitIndex*, only the `oneTimeOnly` *snapshot* action is supported. No update data can be requested or is ever returned for this *snapshot*.

Action request

A *snapshot* action request is used to request LU-related resource data. The action is sent as an m-Action-Confirmed operation.

The *snapshot* request for *logicalUnitIndex* object can specify whether or not a network search is requested for the target name, in addition to looking for matching resources in the VTAM topology agent host node. The search information is specified in the *snapshot* request through values of a managementExtension with the *luSearchParm* parameter where `information` is set to 0 for no-search or 1 for search. Without the managementExtension search parameter, the default is not to perform a network search.

The target resource *logicalUnitIndexName* can be network qualified, such as NETA.APPL1.

The following example shows a request with the target name NETA.APPL1, with "no-search" explicitly specified by the 0 value for `information` in the *luSearchParm* managementExtension.

The following special identifiers are used in the request:

1.3.18.0.0.2291
: *logicalUnitIndex*

1.3.18.0.0.2294
: *logicalUnitIndexName*

1.3.18.0.0.2222
: *snapshot*

1.3.18.0.0.5946
: *luSearchParm*

```
      msg CMIP-1.ROIVapdu (invokeID 196612, operation-value 7,
argument (baseManagedObjectClass 1.3.18.0.0.2291, baseManagedObjectIns
tance (distinguishedName (RelativeDistinguishedName (AttributeValueAss
ertion (attributeType 1.3.18.0.2.4.6, attributeValue "NETA")), Relativ
```

```
eDistinguishedName (AttributeValueAssertion (attributeType 1.3.18.0.0.
2032, attributeValue "SSCP1A")), RelativeDistinguishedName (AttributeV
alueAssertion (attributeType 1.3.18.0.0.2294, attributeValue"NETA.APPL
1")))), actionInfo (actionType 1.3.18.0.0.2222, actionInfoArg (start o
neTimeOnly, addlInfo (ManagementExtension (identifier 1.3.18.0.0.5946,
 significance TRUE, information 0))))))
```

Initial data response

Several objects can be reported for the target name. For example, any of the following combinations are possible:

- A CDRSC might be found on the local VTAM and the corresponding LU might be found on a remote VTAM.
- Both an LU and a USERVAR of the same name might be found on the local VTAM.
- Different LUs with the same name might be found in different networks.

With the `oneTimeOnly` action requested, all the data is returned in linked-replies. Then, to indicate that the data for the entire set of LUs has been returned, the VTAM topology agent sends an additional ROIV linked-reply that is an empty set ROIV followed by an RORS response with only the invoke identifier of the original action request.

Table 20. Reported resources for logicalUnitIndex data

Resource	Object class	Notes
Non-SNA terminal	LU	Local terminal
Application	LU	Model ignored
Dependent LU	LU	
CDRSC	CDRSC	Dynamic alias is ignored. Model ignored.
USERVAR	LUgroup	Report USERVAR name and value
Generic resource	LUgroup	Report generic and real members

If no resource is found for the target name in either the VTAM topology agent host or any remote host being searched, the response is an ROER with error value 1 (*noSuchObjectInstance*).

If a search fails because of a normal VTAM failure to have sessions with a remote host, a processingFailure ROER or a linked-reply ROIV with `specificErrorInfo` of *snaDefinedError* with SNA sense information is sent, possibly after some valid ROIV linked-replies have already been sent for resources found for the target name.

In the following example, an ROER returns SNA sense information because there is no link to the host where the resource was known to be present. (NETAPPL1 has already been reported as a CDRSC owned by SSCP2A).

```
                msg CMIP-1.ROERapdu (invokeID 196610, error-value 10
, parameter (managedObjectClass 1.3.18.0.0.2291, managedObjectInstanc
e (distinguishedName (RelativeDistinguishedName (AttributeValueAssert
ion (attributeType 1.3.18.0.2.4.6, attributeValue "NETA")), RelativeD
istinguishedName (AttributeValueAssertion (attributeType 1.3.18.0.0.2
032, attributeValue "SSCP1A")), RelativeDistinguishedName (AttributeV
alueAssertion (attributeType 1.3.18.0.0.2294, attributeValue "NETAPPL
1")))), specificErrorInfo (errorId 1.3.18.0.0.2266, errorInfo (senseD
ata 087D0001, productIdentification "ACF/VTAM.4.3.0"))))
```

The translated ROER data follows:

```
error-value            : processingFailure
logicalUnitIndexName   : NETA;SSCP1A;NETAPPL1
SpecificErrorInfo
   ErrorID        : snaDefinedError
   ErrorInfo      : senseData 087D0001
                    productIdentification "ACF/VTAM.4.3.0"
```

Action termination

The VTAM topology agent terminates a *snapshot* action for the *logicalUnitIndex* object under the following conditions:

- The requested data has been sent.
- An error occurs during *snapshot* processing in VTAM.
- The association that the *snapshot* is using terminates.

logicalUnitIndex snapshot data

The linked-replies for *logicalUnitIndex* contain data made up of one instance or multiple instances of the following structure:

```
vertex1
       object                       LU-related object distinguished name
       class                        object class
       states                       OSI states for this object
       info
            dependencies               object attribute
            nlrResidentNodePointer     object attribute
            luGroupMembers             object attribute
            cdrscRealLuName            object attribute
            userLabel                  object attribute
```

The following list describes what each field contains.

vertex1
Contains all data reported for a single LU-related object.

class Monitored LU-related objects are reported under the following object classes:

- *logicalUnit*
- *crossDomainResource*
- *luGroup* (for USERVAR or generic resource)

A remote LU that appears to the VTAM topology agent as a CDRSC is reported as an LU under its owning node if the remote LU is found under that node as a result of a network search.

states 14-character string for the following OSI states:

- *operationalState*
- *usageState*
- *administrativeState*
- *availabilityStatus*
- *proceduralStatus*
- *unknownStatus*
- *nativeStatus*

No states information is returned for *luGroup*. No states information is returned if the object was found as the result of a network search.

info Set of attributes for the LU-related object. The info field and the attributes that follow are not included if the object was found as the result of a network search.

Not all attributes are reported for all object classes or are necessarily reported in the order shown in the above structure. The following chart

shows which attributes are possibly reported for an object class.

Table 21. Attributes for logicalUnitIndex reported objects

Attribute	LU	CDRSC	luGroup
dependencies	X	X	
nlrResidentNodePointer		X	
luGroupMembers			X
cdrscRealLuName		X	
userLabel	X	X	

logicalUnitIndex snapshot example

The following example shows the data response for the target name NETA.APPL1. The example includes vertex1 data for one resource found in SSCP1A.

The following special identifiers are used in the response:

1.3.18.0.0.2291
logicalUnitIndex

1.3.18.0.0.2294
logicalUnitIndexName

1.3.18.0.0.2222
snapshot

1.3.18.0.0.2194
dependencies

0.0.13.3100.0.7.50
userLabel

```
                                                    msg CMIP-1.ROIVapdu
(invokeID 131077, linked-ID 196612, operation-value 2, argument (actio
nResult (managedObjectClass 1.3.18.0.0.2291, managedObjectInstance (di
stinguishedName(RelativeDistinguishedName (AttributeValueAssertion (at
tributeType 1.3.18.0.2.4.6, attributeValue"NETA")), RelativeDistinguis
hedName (AttributeValueAssertion (attributeType 1.3.18.0.0.2032, attri
buteValue "SSCP1A")), RelativeDistinguishedName (AttributeValueAsserti
on (attributeType 1.3.18.0.0.2294, attributeValue "NETA.APPL1")))), ac
tionReply (actionType 1.3.18.0.0.2222, actionReplyInfo ((vertex1 (obje
ct (distinguishedName (RelativeDistinguishedName (AttributeValueAssert
ion (attributeType 1.3.18.0.2.4.6, attributeValue "NETA")), RelativeDi
stinguishedName (AttributeValueAssertion (attributeType 1.3.18.0.0.203
2, attributeValue "SSCP1A")), RelativeDistinguishedName (AttributeValu
eAssertion (attributeType 1.3.18.0.0.1984, attributeValue "NETA.APPL1"
)))), class 1.3.18.0.0.1829, states 01000100000000, info (Attribute (a
ttributeId 1.3.18.0.0.2194, attributeValue (dependents (and (Dependent
s (item (distinguishedName (RelativeDistinguishedName (AttributeValueA
ssertion (attributeType 1.3.18.0.2.4.6, attributeValue "NETA")), Relat
iveDistinguishedName (AttributeValueAssertion (attributeType 1.3.18.0.
0.2032, attributeValue "SSCP1A")), RelativeDistinguishedName (Attribut
eValueAssertion (attributeType 1.3.18.0.0.2272, attributeValue "APPL.A
PPL1A"))))), Dependents (item (distinguishedName (RelativeDistinguishe
dName (AttributeValueAssertion (attributeType 1.3.18.0.2.4.6, attribut
eValue "NETA")), RelativeDistinguishedName (AttributeValueAssertion (a
ttributeType 1.3.18.0.0.2032, attributeValue "SSCP1A")))))))), Attrib
ute (attributeId 0.0.13.3100.0.7.50, attributeValue "APPL1")))))))))
```

The following translated initial data shows LU NETA.APPL1 was found in SSCP1A (agent host), under application program major node APPL1A:

```
V1: NETA.SSCP1A.NETA.APPL1(LU)
V1: LU expansion
```

```
vertex 1  : NETA;SSCP1A;NETA.APPL1
          class         : logicalUnit
          states
             Operational State   : Enabled
             Usage State         : Idle
             Administrative State: Unlocked
             Availability Status : No Status
             Procedural Status   : No Status
             Unknown Status      : False
             Native Status       : Active

          info
             userLabel           : APPL1
             dependencies        : NETA;SSCP1A;APPL.APPL1A
                                   NETA.SSCP1A
```

Appendix A. C language header file (ACYAPHDH)

The following C language header file, ACYAPHDH, contains many declarations that are needed for compiling a C program that uses the CMIP services API. This header file is shipped in the AMACLIB data set of the SYS1.MACLIB data set.

```
/*                                                                  */ 00050000
/*   COPYRIGHT = LICENSED MATERIALS - PROPERTY OF IBM               */ 00100000
/*                                                                  */ 00150000
/*               THIS PRODUCT CONTAINS                              */ 00200000
/*               "RESTRICTED MATERIALS OF IBM"                      */ 00250000
/*                                                                  */ 00300000
/*               5695-117 (C) COPYRIGHT IBM CORP. 1994              */ 00350000
/*               ALL RIGHTS RESERVED.                               */ 00400000
/*                                                                  */ 00450000
/*               U.S. GOVERNMENT USERS RESTRICTED RIGHTS -          */ 00500000
/*               USE, DUPLICATION OR DISCLOSURE RESTRICTED          */ 00550000
/*               BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.        */ 00600000
/*                                                                  */ 00650000
/*               SEE COPYRIGHT INSTRUCTIONS.                        */ 00700000
/*                                                                  */ 00750000
/* $MAC(ACYAPHDH),COMP(CMIP),PROD(VTAM): CMIP MIB API declarations  */ 00800000
/*                                                                  */ 00850000
/* FLAG REASON   RELEASE DATE   ORIGIN  FLAG DESCRIPTORS            */ 00900000
/* ---- -------- ------- ------ ------  ---------------------------- */ 00950000
/* $L0= FYJDR002 VTAGENT 940523 647877: VTAM Agent                  */ 01000000
/* $Y1= P115000  VTAGENT 940801 792173: Add Address Fields to end   */ 01050000
/*                                      of Vector                   */ 01100000
/* $Y2= P115922  VTAGENT 940810 647877: Fix declaration of          */ 01150000
/*                                      MIBSendDeleteRegistration   */ 01200000
/*                                                                  */ 01250000
                                                                      01300000
#ifndef ACYAPHDH_INCLUDED           /* Only process these once.     */ 01350000
                                                                      01400000
#define ACYAPHDH_INCLUDED           /* Signify that these have been    01450000
                                       processed.                   */ 01500000
                                                                      01550000
#include                    /* Obtain definition of time_t for  01600000
                                       APIhdr.                      */ 01650000
                                                                      01700000
/*******************************************************************/ 01750000
/* The following constant is the maximum number of invoke ids which  */ 01800000
/* may concurrently active on the same connection.  It is the        */ 01850000
/* maximum allowed value for the max outstanding invoke ids          */ 01900000
/* parameter on MIBConnect.                                          */ 01950000
/*******************************************************************/ 02000000
                                                                      02050000
#define INVOKE_ID_MAX  0x00010000                                     02100000
                                                                      02150000
/*******************************************************************/ 02200000
/* The following constant is the length in bytes of the longest      */ 02250000
/* possible local identifier.                                        */ 02300000
/*******************************************************************/ 02350000
                                                                      02400000
#define LOCAL_ID_MAX   8                                              02450000
                                                                      02500000
/*******************************************************************/ 02550000
/* The following constants represent the settings of the msg_type    */ 02600000
/* field in the APIhdr structure.                                    */ 02650000
/*******************************************************************/ 02700000
                                                                      02750000
```

```
#define API_MSG                0                                        02800000
#define API_REG_ACCEPT         1                                        02850000
#define API_SVC_COMPLETE       2                                        02900000
#define API_SVC_ERROR          3                                        02950000
#define API_TERMINATE_INSTANCE  4                                       03000000
                                                                        03050000
/*********************************************************************/ 03100000
/* The following constants represent the settings of the origin      */ 03150000
/* field in the APIhdr structure.                                    */ 03200000
/*********************************************************************/ 03250000
                                                                        03300000
#define ORIGIN_OBJ       0          /* The request which initiated      03350000
                                       this message was generated by    03400000
                                       the object which received this   03450000
                                       message.                      */ 03500000
                                                                        03550000
#define ORIGIN_REMOTE    1          /* The request which initiated      03600000
                                       this message was generated by    03650000
                                       an object other than the one     03700000
                                       which received this message      03750000
                                       (unless the object generated a   03800000
                                       request to itself).           */ 03850000
                                                                        03900000
/*********************************************************************/ 03950000
/* The following constants are used for the connection options       */ 04000000
/* parameter of MIBConnect.                                          */ 04050000
/*********************************************************************/ 04100000
                                                                        04150000
#define NO_CONNECT_OPTIONS     0                                        04200000
#define SHORT_NAMES            2                                        04250000
                                                                        04300000
/*********************************************************************/ 04350000
/* The following constants represent valid capabilities bits which   */ 04400000
/* can be specified in the capabilities parameter of MBReg().        */ 04450000
/*********************************************************************/ 04500000
                                                                        04550000
#define NO_CAPABILITIES  0          /* no special capabilities       */ 04600000
                                                                        04650000
#define SUBTREE_MANAGER  1          /* The object being registered is   04700000
                                       a subtree manager.            */ 04750000
                                                                        04800000
/*********************************************************************/ 04850000
/* The following constant is used as the value of the name type      */ 04900000
/* parameter of MIBSendRegister.                                     */ 04950000
/*********************************************************************/ 05000000
                                                                        05050000
#define DN_OF_INSTANCE   0                                              05100000
                                                                        05150000
/*********************************************************************/ 05200000
/* The following constants represent values for the dest type        */ 05250000
/* parameter of MIBSendCmipRequest.                                  */ 05300000
/*********************************************************************/ 05350000
                                                                        05400000
#define DS_NOT_PROVIDED       0                                         05450000
#define DS_FULL_DN            1                                         05500000
#define DS_ASSOC_HANDLE       2                                         05550000
#define DS_AE_TITLE           3                                         05600000
                                                                        05650000
/*********************************************************************/ 05700000
/* Structures                                                        */ 05750000
/*********************************************************************/ 05800000
                                                                        05850000
typedef struct APIhdr_tag                                               05900000
{                                                                       05950000
  unsigned char msg_type;                                               06000000
  unsigned char api_version;                                            06050000
  unsigned char origin;                                                 06100000
```

```
  unsigned char RESERVED1;          /* Applications must not use or     06150000
                                       depend on the value of this      06200000
                                       field in any way.             */ 06250000
  unsigned int invokeId;                                                06300000
  unsigned int connectId;                                               06350000
  unsigned int numLocalIds;                                             06400000
  time_t timestamp;                                                     06450000
  unsigned short resultCode;                                            06500000
  unsigned char RESERVED2??(2??);   /* Applications must not use or     06550000
                                       depend on the value of this      06600000
                                       field in any way.             */ 06650000
  unsigned int  RESERVED3;          /* Applications must not use or     06700000
                                       depend on the value of this      06750000
                                       field in any way.             */ 06800000
  unsigned char localIds??(8??);                                        06850000
} APIhdr;                                                               06900000
                                                                        06950000
/*********************************************************************/ 07000000
/* The following structures are used by CMIP applications using      */ 07050000
/* Data Spaces:                                                      */ 07100000
/*    (1) DataSpaceVector Format  (ISTRIV10_t)                       */ 07150000
/*    (2) Interface Control Block (ISTNMICB_t)                       */ 07200000
/*********************************************************************/ 07250000
                                                                        07300000
typedef struct ISTRIV10_tag                                             07350000
{                                                                       07400000
  char  RIV10LEN;                   /* VECTOR LENGTH                */ 07450000
  char  RIV10KEY;                   /* VECTOR KEY                   */ 07500000
  char  RIV10DSN??(8??);            /* DATA SPACE NAME              */ 07550000
  char  RIV10NMI??(4??);            /* ADDRESS OF ISTNMICB IN DATA     07600000
                                       SPACE STORAGE                */ 07650000
  char  RIV10STK??(8??);            /* STOKEN OF DATA SPACE         */ 07700000
  char  reserved1??(4??);           /* reserved - not available     */ 07750000
  char  RIV10CDQ??(4??);            /* Address of Dequeue Routine   */ 07800000
  char  RIV10CRL??(4??);            /* Address of Release Routine   */ 07850000
  char  reserved2??(8??);           /* reserved                     */ 07900000
} ISTRIV10_t;                                                           07950000
                                                                        08000000
typedef struct ISTNMICB_tag                                             08050000
{                                                                       08100000
  char reserved1??(4??);            /* reserved - not available    */  08150000
  void *NMIPDCDQ;                   /* Address of Dequeue Routine  */  08200000
  void *NMIPDCRL;                   /* Address of Release Routine  */  08250000
} ISTNMICB_t;                                                           08300000
                                                                        08350000
/*********************************************************************/ 08400000
/* The following macro can be used to calculate the size of a given  */ 08450000
/* APIhdr (including any local identifiers present).                 */ 08500000
/*********************************************************************/ 08550000
                                                                        08600000
#define APIhdrSize(x,size)                          \                   08650000
        ((char *)&((x).localIds) - (char *)&(x) +   \                   08700000
        (size) * (x).numLocalIds)                                       08750000
                                                                        08800000
/*********************************************************************/ 08850000
/* The following type definitions are provided so that the C         */ 08900000
/* compiler can perform type checking on the MIB API calls.  The     */ 08950000
/* address of each MIB API routine should be given one of the        */ 09000000
/* following types.                                                  */ 09050000
/*********************************************************************/ 09100000
                                                                        09150000
typedef int MIBConnect_t(                                               09200000
               unsigned int,        /* API level                    */ 09250000
               int *,               /* link id                      */ 09300000
               unsigned int,        /* max outstanding invoke          09350000
                                       ids                          */ 09400000
               const char *,        /* application name             */ 09450000
```

```
                void *,             /* TPEND exit                    */ 09500000
                void *,             /* read queue exit               */ 09550000
                unsigned int *,     /* SMAE name buffer size         */ 09600000
                char *,             /* SMAE name buffer              */ 09650000
                unsigned int *,     /* System Object name buffer        09700000
                                       size                          */ 09750000
                char *,             /* System Object name            */ 09800000
                int,                /* user data field               */ 09850000
                unsigned int *,     /* error flag                    */ 09900000
                char **,            /* VTAM release level            */ 09950000
                const char *,       /* password                      */ 10000000
                unsigned int,       /* length of DS vector           */ 10050000
                ISTRIV10_t *,       /* DS vector                     */ 10100000
                unsigned int,       /* local identifier length       */ 10150000
                unsigned int);      /* connection options            */ 10200000
                                                                        10250000
#pragma linkage(MIBConnect_t,OS)                                        10300000
                                                                        10350000
typedef int MIBDisconnect_t(                                            10400000
                int,                /* link id                       */ 10450000
                unsigned int *);    /* return error flag             */ 10500000
                                                                        10550000
#pragma linkage(MIBDisconnect_t,OS)                                     10600000
                                                                        10650000
typedef int MIBSendRegister_t(                                          10700000
                int,                /* link id                       */ 10750000
                unsigned int *,     /* returned invoke id            */ 10800000
                const void *,       /* local id                      */ 10850000
                const char *,       /* object class                  */ 10900000
                int,                /* name type                     */ 10950000
                const char *,       /* distinguished name            */ 11000000
                const char *,       /* name binding oid              */ 11050000
                unsigned int,       /* capability flags              */ 11100000
                unsigned int,       /* allomorphs count              */ 11150000
                char **,            /* allomorphs array              */ 11200000
                unsigned int,       /* create handlers count         */ 11250000
                char **);           /* create handlers array         */ 11300000
                                                                        11350000
#pragma linkage(MIBSendRegister_t,OS)                                   11400000
                                                                        11450000
typedef int MIBSendDeleteRegistration_t(                                11500000
                int,                /* link id                       */ 11550000
                unsigned int *,     /* returned invoke id            */ 11600000
                const void *,       /* local identifier              */ 11650000
                const char *);      /* DN                         @Y2C*/ 11700000
                                                                        11750000
#pragma linkage(MIBSendDeleteRegistration_t,OS)                         11800000
                                                                        11850000
typedef int MIBSendRequest_t(                                           11900000
                int,                /* link id                       */ 11950000
                unsigned int *,     /* returned invoke id            */ 12000000
                const void *,       /* local identifier              */ 12050000
                const char *);      /* message                       */ 12100000
                                                                        12150000
#pragma linkage(MIBSendRequest_t,OS)                                    12200000
                                                                        12250000
typedef int MIBSendResponse_t(                                          12300000
                int,                /* link id                       */ 12350000
                unsigned int,       /* invoke id                     */ 12400000
                const void *,       /* local identifier              */ 12450000
                const char *,       /* source                        */ 12500000
                const char *,       /* dest association handle       */ 12550000
                const char *);      /* message                       */ 12600000
                                                                        12650000
#pragma linkage(MIBSendResponse_t,OS)                                   12700000
                                                                        12750000
typedef int MIBSendCmipRequest_t(                                       12800000
```

```
              int,                /* link id                      */ 12850000
              unsigned int,       /* argument type                */ 12900000
              const char *,       /* argument                     */ 12950000
              const void *,       /* local identifier             */ 13000000
              const char *,       /* source                       */ 13050000
              unsigned int,       /* type of destination          */ 13100000
              const char *,       /* destination                  */ 13150000
              unsigned int *);    /* returned invoke id           */ 13200000
                                                                     13250000
#pragma linkage(MIBSendCmipRequest_t,OS)                             13300000
                                                                     13350000
typedef int MIBSendCmipResponse_t(                                   13400000
              int,                /* link id                      */ 13450000
              unsigned int,       /* invoke id                    */ 13500000
              unsigned int,       /* last in chain?               */ 13550000
              unsigned int,       /* success?                     */ 13600000
              unsigned int,       /* argument type                */ 13650000
              const char *,       /* argument                     */ 13700000
              const void *,       /* local identifier             */ 13750000
              const char *,       /* source                       */ 13800000
              const char *,       /* dest association handle      */ 13850000
              unsigned int *);    /* returned invoke id           */ 13900000
                                                                     13950000
#pragma linkage(MIBSendCmipResponse_t,OS)                            14000000
                                                                     14050000
/*********************************************************************/ 14100000
/*  The following constants are for the synchronous return codes    */ 14150000
/*  which may be received from one of the MIB API routines or via   */ 14200000
/*  an API_SVC_ERROR message from CMIP Services.                    */ 14250000
/*********************************************************************/ 14300000
                                                                     14350000
#define MB_ERR_ALLOC                              7                  14400000
#define MB_ERR_MAX_OUTSTANDING                  932                  14450000
                                                                     14650000
/*********************************************************************/ 14700000
/*  The following constants are for the synchronous return codes    */ 14750000
/*  which may be received only from one of the MIB API routines.    */ 14800000
/*********************************************************************/ 14850000
                                                                     14900000
#define MB_ERR_INVALID_LINK_ID                    918                14946800
#define MB_ERR_NOT_REGISTERED                     920                14993600
#define MB_ERR_CONNECT                            945                15040400
#define MB_WARN_DATA_SPACE_FULL                  1000                15087200
#define MB_WARN_EXIT_FAILURE                     1001                15134000
#define MB_DATA_ON_DATA_SPACE                    1002                15180800
#define MB_ERR_INVALID_ENVIRONMENT               1003                15227600
#define MB_ERR_INVALID_ARGUMENT                  1004                15274400
#define MB_ERR_INVALID_ARGUMENT_TYPE             1005                15321200
#define MB_ERR_INVALID_ASSOC_HANDLE              1006                15368000
#define MB_ERR_INVALID_SMAE_NAME                 1007                15414800
#define MB_ERR_CMIP_SERVICES_INACTIVE            1008                15461600
#define MB_ERR_INVALID_DS_VECTOR                 1009                15508400
#define MB_ERR_INVALID_DEST_TYPE                 1010                15555200
#define MB_ERR_INVALID_DIST_NAME                 1011                15602000
#define MB_ERR_INVALID_MAX_INVOKE_IDS            1012                15648800
#define MB_ERR_INVALID_API_LEVEL                 1013                15695600
#define MB_ERR_INVALID_APPL_NAME                 1014                15742400
#define MB_ERR_INVALID_DS_VECTOR_SIZE            1015                15789200
#define MB_ERR_INVALID_SMAE_NAME_SIZE            1016                15836000
#define MB_ERR_INVALID_INVOKE_ID                 1017                15882800
#define MB_ERR_MIBDISCONNECT                     1018                15929600
#define MB_ERR_INVALID_MSG                       1019                15976400
#define MB_ERR_INVALID_OBJECT_CLASS              1020                16023200
#define MB_ERR_INVALID_READ_QUEUE_EXIT           1021                16070000
#define MB_ERR_INVALID_SYSTEM_NAME_SIZE          1022                16116800
#define MB_ERR_INVALID_LOCAL_ID_SIZE             1023                16163600
#define MB_ERR_TRANSMIT                          1024                16210400
```

```
#define MB_ERR_VTAM_INACTIVE                    1025                16257200
#define MB_ERR_INVALID_USER_DATA                1026                16304000
#define MB_ERR_INVALID_ERROR_FLAG               1027                16350800
#define MB_ERR_INVALID_RELEASE_LEVEL            1028                16397600
#define MB_ERR_INVALID_PASSWORD                 1029                16444400
#define MB_ERR_INVALID_CAPABILITY_FLAGS         1030                16491200
#define MB_ERR_INVALID_TPEND_EXIT               1031                16538000
#define MB_ERR_INVALID_LAST_IN_CHAIN_FLAG       1032                16584800
#define MB_ERR_INVALID_SUCCESS_FLAG             1033                16631600
#define MB_ERR_INVALID_SYSTEM_NAME              1034                16678400
#define MB_ERR_INVALID_CONNECT_OPTIONS          1035                16725200
#define MB_ERR_INVALID_NAME_TYPE                1036                16772000
#define MB_ERR_INVALID_NAME_BINDING             1037                16818800
#define MB_ERR_INVALID_ALLOMORPHS_COUNT         1038                16865600
#define MB_ERR_INVALID_ALLOMORPHS_ARRAY         1039                16912400
#define MB_ERR_INVALID_CREATE_HANDLERS_COUNT    1040                16959200
#define MB_ERR_INVALID_CREATE_HANDLERS_ARRAY    1041                17006000
#define MB_ERR_INVALID_LOCAL_ID                 1042                17052800
#define MB_ERR_INVALID_DEST                     1043                17099600
                                                                    17150000
/*********************************************************************/ 17200000
/* The following return codes are returned from CMIP Services only   */ 17250000
/* via API_SVC_ERROR messages.                                       */ 17300000
/*********************************************************************/ 17350000
                                                                    17400000
#define PROGRAM_CHECK                             8                 17433300
                                                                    17466600
#define AUTHENTICATION_FAILED                   250                 17499900
#define AUTHENTICATION_INFO_MISSING             251                 17533200
#define AUTHENTICATION_MECH_UNKNOWN             252                 17566500
                                                                    17600000
#define BER_BAD_TYPE                            300                 17650000
#define BER_BAD_MODULE                          301                 17700000
#define BER_NULL_TYPE                           302                 17750000
#define BER_NULL_MODULE                         303                 17800000
#define BER_NULL_STRING                         304                 17850000
#define BER_NULL_STRUCT                         305                 17900000
#define BER_BAD_METATABLE                       306                 17950000
#define BER_UNKNOWN_TYPE                        307                 18000000
#define BER_UNKNOWN_MEMBER                      308                 18050000
#define BER_UNKNOWN_ALTERNATIVE                 309                 18100000
#define BER_NO_END_PARENTHESIS                  310                 18150000
#define BER_NO_START_PARENTHESIS                311                 18200000
#define BER_NO_MORE_STRING                      312                 18250000
#define BER_PARSE_ERROR                         313                 18300000
#define BER_IMPLICIT_CHOICE                     314                 18350000
#define BER_CANNOT_RESOLVE                      315                 18400000
#define BER_NEED_LABEL                          316                 18450000
#define BER_MISSING_MEMBER                      317                 18500000
#define BER_NO_PARENT                           319                 18550000
#define BER_BAD_DN_PARSE                        320                 18600000
#define BER_BAD_RESOLUTION_NODE                 321                 18650000
#define BER_MISSING_RESOLUTION_NODE             322                 18700000
#define BER_LABEL_MISMATCH                      323                 18750000
#define BER_NOT_BOOLEAN                         325                 18800000
#define BER_NOT_INTEGER                         326                 18850000
#define BER_NOT_REAL                            327                 18900000
#define BER_NOT_NULL                            328                 18950000
#define BER_NOT_BIT_STRING                      329                 19000000
#define BER_NOT_HEX_STRING                      330                 19050000
#define BER_BAD_HEX_STRING                      331                 19100000
#define BER_NOT_OI                              332                 19150000
#define BER_BAD_TIME                            333                 19200000
#define BER_BAD_ENUMERATED                      334                 19250000
#define BER_BAD_PRINTABLE_STRING                335                 19300000
#define BER_BAD_NUMERIC_STRING                  336                 19350000
#define BER_BAD_VISIBLE_STRING                  337                 19400000
```

```
#define BER_BAD_GRAPHIC_STRING                  338                 19450000
#define BER_BAD_GENERAL_STRING                  339                 19500000
#define BER_BAD_IA5_STRING                      340                 19550000
#define BER_DUPLICATE_MEMBER                    341                 19600000
#define BER_CANT_DO_REAL                        342                 19650000
#define BER_NOT_STRAIGHT_BER                    343                 19700000
#define BER_UNRESOLVED_EXTERNAL                 344                 19750000
#define BER_STILL_MORE_STRING                   345                 19800000
#define BER_DUP_MODULE                          347                 19850000
#define BER_UNRESOLVED_MODULE_REF               348                 19900000
#define BER_UNRESOLVED_REF                      349                 19950000
#define BER_FAILED_SUBTYPE                      354                 20000000
#define BER_BAD_CONSTRUCTED                     356                 20050000
#define BER_BAD_PRIMITIVE                       357                 20100000
#define BER_BAD_INITIAL_OCTET                   358                 20150000
#define BER_BAD_BOOLEAN                         359                 20200000
#define BER_BAD_OI                              360                 20250000
#define BER_BAD_NULL                            361                 20300000
#define BER_BAD_PARAMETERS                      363                 20400000
#define BER_EMPTY_BIT_STRING                    364                 20450000
                                                                    20800000
#define RDN_SEP_AT_BEGIN_OF_DN                  375                 20847600
#define AVA_SEP_AT_BEGIN_OF_DN                  376                 20895200
#define SPACE_AT_BEGIN_OF_DN                    377                 20942800
#define INVALID_CHAR_AT_BEGIN_OF_DN             378                 20990400
#define RDN_SEP_AT_BEGIN_OF_RDN                 379                 21038000
#define AVA_SEP_AT_BEGIN_OF_RDN                 380                 21085600
#define SPACE_AT_BEGIN_OF_RDN                   381                 21133200
#define INVALID_CHAR_AT_BEGIN_OF_RDN            382                 21180800
#define INVALID_ALPHA_IN_INTEGER_VALUE          383                 21228400
#define INVALID_SPACE_IN_INTEGER_VALUE          384                 21276000
#define INVALID_CHAR_IN_INTEGER_VALUE           385                 21323600
#define INVALID_SPACE_IN_OI_VALUE               386                 21371200
#define INVALID_CHAR_IN_OI_VALUE                387                 21418800
#define INVALID_SPACE_IN_SYMBOLIC_VALUE         388                 21466400
#define INVALID_CHAR_IN_SYMBOLIC_VALUE          389                 21514000
#define INVALID_CHAR_IN_ATTR_VALUE              390                 21561600
#define INVALID_SPACE_IN_ATTR_VALUE             391                 21609200
#define PREMATURE_END_OF_DN                     392                 21656800
#define INVALID_SPACE_AT_END_OF_RDN             393                 21704400
#define BOTH_QUOTE_TYPES_USED                   394                 21752000
                                                                    21800000
#define REPL_ERR_INVLD_VERBCODE                 400                 21850000
#define REPL_ERR_MISSING_ASN1_TREE              401                 21900000
#define REPL_ERR_OBJCLASS_MISSING               402                 21950000
#define REPL_ERR_OBJCLASS_INVALID               403                 22000000
#define REPL_ERR_OBJINST_MISSING                404                 22050000
#define REPL_ERR_OBJINST_INVALID                405                 22100000
#define REPL_ERR_DUPLICATE_OBJINST              406                 22150000
#define REPL_ERR_NO_SUCH_OBJINST                407                 22200000
#define REPL_ERR_MOI_OC_MISMATCH                408                 22250000
#define REPL_ERR_NAME_CREATE_FAILED             409                 22300000
#define REPL_ERR_GDMO_FILE_BAD_VERS             410                 22350000
#define REPL_ERR_NOTHING_TO_DELETE              411                 22400000
#define REPL_WRN_OBJCLASS                       412                 22450000
#define REPL_ERR_ALREADY_AN_STM                 413                 22500000
#define REPL_ERR_INVLD_STM_CHILD                414                 22550000
#define REPL_ERR_SCOPES_TO_NOTHING              415                 22600000
#define REPL_ERR_INVALID_SCOPE                  416                 22650000
#define REPL_ERR_COMMITDN_NOTIN_LIST            417                 22700000
#define REPL_ERR_NO_ONE_2_SEND_CRT_2            419                 22750000
#define REPL_ERR_NOONE_2_SEND_EVNT_2            420                 22800000
#define REPL_ERR_ALREADY_EVNT_HNDLR             421                 22850000
#define REPL_ERR_NAMEBIND_INVALID               422                 22900000
#define REPL_ERR_CRT_FAIL_NB                    424                 22950000
#define REPL_ERR_CRT_FAIL_NO_NB                 425                 23000000
#define REPL_ERR_DLT_FAIL_CONTOBJS              426                 23050000
```

```
#define REPL_ERR_DLT_FAIL_TO_DCO                427                 23100000
#define REPL_ERR_DLT_FAIL_NB                    428                 23150000
#define REPL_ERR_NO_LOCALDN                     429                 23200000
#define REPL_ERR_DUPLICATE_LDNH                 430                 23250000
#define REPL_REG_CREATED                        431                 23300000
#define REPL_CRT_COMPLETED                      432                 23350000
#define REPL_REG_COMPLETED                      433                 23400000
#define REPL_REG_SUSPENDED                      434                 23450000
#define REPL_ERR_ATTRTYPE_MISMATCH              435                 23500000
#define REPL_ERR_CANNOT_CHANGE_NB               436                 23550000
#define REPL_ERR_BULK_HAD_PROBLEMS              438                 23600000
#define REPL_ERR_NB_DISALLOWS_NEWOC             439                 23650000
#define REPL_ERR_SYNC_NOT_SUPPORTED             440                 23700000
                                                                    23800000
#define CRC_ERR_INVLD_VERBCODE                  500                 23836300
#define CRC_ERR_INVLD_SESSHAND                  501                 23872600
#define CRC_ERR_INVLD_INVOKEID                  502                 23908900
#define CRC_ERR_DPLCT_INVOKEID                  503                 23945200
#define CRC_ERR_INVLD_LINKEDID                  504                 23981500
#define CRC_ERR_UNABLE_TO_BUILD_MSG             505                 24017800
#define CRC_ERR_INVLD_ROERRJ_RCVD               506                 24054100
#define CRC_ERR_INVLD_CANCELGET                 507                 24090400
#define CRC_ERR_INVLD_INVOKEID_ON_CANCELGET     508                 24126700
#define CRC_DELETE_RORJ_RECEIVED                509                 24163000
                                                                    24200000
#define SSERR_STATE_INVALID                     550                 24250000
#define SSERR_SPDU_INVALID                      551                 24300000
#define SSERR_MISSING_PI                        552                 24350000
#define SSERR_MISSING_UI                        553                 24400000
#define SSERR_VERB_INVALID                      554                 24450000
#define SSERR_INVALID_SUR                       556                 24500000
#define SSERR_USERDATA_SIZE                     557                 24550000
#define SSERR_TDISC_UNSPECIFIED                 558                 24600000
#define SSERR_TDISC_CONGESTED                   559                 24650000
#define SSERR_TDISC_UNATTACHED                  560                 24700000
#define SSERR_TDISC_ADDRESS                     561                 24750000
#define SSERR_VERSION                           562                 24800000
#define SSERR_PARTNER_ABORT                     563                 24850000
#define SSERR_ENCLOSURE_ITEM                    564                 24900000
                                                                    25000000
#define MD_ERR_BAD_MDSMU                        568                 25050000
#define MD_ERR_SNACR_BEING_SENT                 573                 25100000
#define MD_ERR_SNACR_RECEIVED                   574                 25150000
                                                                    25200000
#define SSERR_GIVE_TOKEN_NO_DATA                578                 25250000
#define SSERR_DUPLICATE                         581                 25300000
                                                                    25750000
#define ACF_EVENT_LOOP                          802                 25789300
#define ACF_INVALID_ASSOC_ID                    803                 25828600
#define ACF_INVALID_PARAMETERS                  804                 25867900
#define ACF_INVALID_USER_ID                     806                 25907200
#define ACF_RSP_BUILD_SEND_FAILED               807                 25946500
#define ACF_ERR_KILL_LOC_ASSOC                  808                 25985800
#define ACF_UNSUPPORTED_VERB                    809                 26025100
#define ACF_INVALID_DEST_FORMAT                 810                 26064400
#define ACF_BAD_AE_TITLE_FORMAT                 812                 26103700
#define ACF_CANNOT_FIND_INST                    814                 26143000
#define ACF_NO_DESTINATION                      815                 26182300
#define ACF_NO_ASSOC_TEMP                       817                 26221600
#define ACF_MSG_REJECTED                        818                 26260900
#define ACF_EMPTY_DEF_LIST_RESULT               819                 26300200
#define ACF_QUEUED_MESSAGE                      823                 26339500
#define ACF_ASSOC_ID_WRAP                       824                 26378800
#define ACF_AUTO_ASSOC_TEARDOWN_TERM            825                 26418100
#define ACF_TOO_MANY_LOCAL_ASSOCS               826                 26457400
#define ACF_DUPLICATE_AE                        827                 26496700
#define ACF_REMOTE_AE                           828                 26536000
```

```
#define ACF_INVALID_STATE_TO_RELEASE              829                 26575300
#define ACF_INVALID_AE                            830                 26614600
#define ACF_BAD_P_MODE                            831                 26653900
#define ACF_BAD_P_PROTOCOL_VERSION                832                 26693200
#define ACF_BAD_CMIP_VERSION                      833                 26732500
#define ACF_BAD_APPL_CONTEXT                      834                 26771800
#define ACF_NO_APPL_CONTEXT                       835                 26811100
#define ACF_NO_APPL_INFO                          836                 26850400
#define ACF_BAD_DEF_LIST                          837                 26889700
#define ACF_WRONG_AE_TITLE                        838                 26929000
#define ACF_ALREADY_CONFIRMED                     839                 26968300
#define ACF_NO_AE_QUALIFIER                       840                 27007600
                                                                      27050000
#define MB_ERR_PROCFAIL_NOT_OPTIONAL              900                 27100000
#define MB_ERR_COMPXLIM_NOT_OPTIONAL              901                 27150000
#define MB_ERR_INVALID_TYPENAME                   903                 27200000
#define MB_ERR_NOT_CONNECTED                      904                 27250000
#define MB_ERR_WRONG_LENGTH                       906                 27300000
#define MB_ERR_INVALID_API_TAG                    907                 27350000
#define MB_ERR_MISSING_API_TAG                    909                 27400000
#define MB_ERR_UNSUPPORTED_MSG_TYPE               913                 27450000
#define MB_ERR_DELETE_PROTOCOL_ERROR              914                 27500000
#define MB_ERR_DUPLICATE_TAGS                     915                 27550000
#define MB_ERR_CONNECTION_CONFLICT                916                 27600000
#define MB_ERR_HEADER_NOT_PRESENT                 917                 27650000
#define MB_ERR_INVALID_STATE                      919                 27700000
#define MB_ERR_CMIP_ERR_RESP_ILLEGAL              921                 27750000
#define MB_ERR_CMIP_ERR_RESP_STKCHK               922                 27800000
#define MB_TRY_XMIT_RESP                          923                 27850000
#define MB_ERR_LOST_CONNECTION                    925                 27900000
#define MB_ERR_INVALID_NUMLOCALIDS                926                 27950000
#define MB_ERR_MISSING_LOCAL_ID                   928                 28000000
#define MB_ERR_LOCAL_ID_ALREADY_REGISTERED        929                 28050000
#define MB_ERR_INVALID_MSG_TYPE                   930                 28150000
#define MB_ERR_SOURCE_NOT_IN_SUBTREE              931                 28200000
#define MB_ERR_CMIP_ERR_NOT_STM                   933                 28250000
#define MB_ERR_NOT_SUBTREE_MGR                    934                 28300000
#define MB_ERR_DIDNT_USE_AMPER_IID                935                 28350000
#define MB_ERR_CMIP_ERR_NOTASROIV                 936                 28400000
#define MB_ERR_INVALID_MSG_FORMAT                 937                 28450000
#define MB_ERR_EMPTY_ROIV_INVALID                 938                 28500000
#define MB_ERR_INVALID_RESP                       939                 28550000
#define MB_ERR_CANCELGET_RESP_INVALID             941                 28600000
                                                                      28650000
#define HDR_SYNTAX_ERROR                          952                 28700000
#define INVALID_HDR_DEST_TYPE                     953                 28750000
#define INVALID_HDR_SRC_TYPE                      954                 28800000
#define UNRECOGNIZED_HDR_LABEL                    955                 28850000
#define KEY_IS_NULL                               956                 28900000
#define KEY_NOT_FOUND                             957                 28950000
#define MIB_VAR_NOT_LOADED                        958                 29000000
                                                                      29100000
#define LABV_END_QUOTE_NOT_FOUND                  961                 29150000
#define LABV_NULL_VALUE                           962                 29200000
#define LABV_INVALID_CHAR_IN_VALUE                963                 29250000
#define LABV_INVALID_GROUP_DELIMITER              964                 29300000
#define LABV_EMPTY_STRING                         965                 29350000
                                                                      29400000
#endif                              /* ifndef ACYAPHDH_INCLUDED      */ 29450000
```

Appendix B. ASN.1 specification of the basic CMIP strings

The following ASN.1 syntax is contained in the CMIP-1 ASN.1 module of the ACYIDCMS member of the SYS1.SISTASN1 data set.

```
--
--   COPYRIGHT = LICENSED MATERIALS - PROPERTY OF IBM
--
--               THIS PRODUCT CONTAINS
--               "RESTRICTED MATERIALS OF IBM"
--
--               5695-117 (C) COPYRIGHT IBM CORP. 1994
--               ALL RIGHTS RESERVED.
--
--               U.S. GOVERNMENT USERS RESTRICTED RIGHTS -
--               USE, DUPLICATION OR DISCLOSURE RESTRICTED
--               BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
--
--               SEE COPYRIGHT INSTRUCTIONS.
--
-- Modules: CMIP-1 SMASE-A_ASSOCIATE-Information CMIP-A-Associate-Information CMIP-A-Abort-Information
-- Based on standard:  CMIP
--     CMIP - based on ISO/IEC 9596-1 dated 24 November 1990
--         Modules: CMIP-A-ASSOCIATE-Information, CMIP-A-ABORT-Information, CMIP-1
--     SMO - based on ISO/IEC IS 10040 dated August 1991
--         Module:  SMASE-A-ASSOCIATE-Information
--
-- ____________________________________________________________________________
--|
--|  This copy of the CMIP ASN.1 definitions has been adapted for use with
--|  cmipWorks. It eliminates the CHOICE from the definitions of AttributeId,
--|  ActionTypeId and EventTypeId. The resulting syntax will encode and flow
--|  IDENTICALLY to the original productions.
--|
--|  This change was made because:
--|    It simplifies the processing of all messages  - eliminating the useless
--|       CHOICE that would need to be processed on every message.
--|    It simplifies the interface used by applications to avoid writing
--|             (globalForm 1.2.3) for every attribute, action and event.
--|    It aligns with the actual use of the fields since they cannot use the
--|       localForm unless the application context specifically allocates values
--|       for it and assigns the correspondence to data types.
--|
-- |____________________________________________________________________________
--
--
--
--
-- Differences from standard:
--    - Modify the definition of access control information to allow
--      the correct thing to be parsed. Replace the OCTET STRING with
--      a cobbled-up External.
--    - Various comments have been added for explanations, and for use by tools
--    - Module SMASE-A-ASSOCIATE-Information has an errant comma in the standard after agentRoleFunctionalUnit
--       causing the module not to parse.  The problem has been corrected in this version
--    - CMIP-A-ASSOCIATE-Information
--      - None
--    - CMIP-1:
--      - IMPORTS and EXPORTS do no match standard
--      - Remote-Operations-APDUs is loosely translated as part of CMIP-1.  The differences from the
--        RO standards are that the ROIV, RORS, ROER, and RORJ definitions are tagged and IMPLICIT;
--        SEQUENCE in RORSapdu has a label
--        macros are not used     *** RO also in module Remote-Operations-APDUs ***
--      - ActionArgument... COMPONENTS OF... -> ObjectClass and ObjectInstance due to parser limitation
--      - ActionTypeId is defined as a CHOICE in the standards - here it is only an OI (see in line
--        comments
--      - AttributeId is defined as a CHOICE in the standards - here it is only an OI
--      - DeleteArgument... COMPONENTS OF... -> ObjectClass and ObjectInstance due to parser limitation
--      - EventTypeId is defined as a CHOICE in the standards - here it is only an OI
--      - GetArgument... COMPONENTS OF... -> ObjectClass and ObjectInstance due to parser limitation
--      - Notification added to support allomorphic Notifications (see in line comments)
--      - ObjectClass is defined as a CHOICE in the standards - here it is only an OI
--      - basicScope label put on first choice of Scope
--      - SetArgument... COMPONENTS OF... -> ObjectClass and ObjectInstance due to parser limitation
--

SMASE-A-ASSOCIATE-Information
  {joint-iso-ccitt ms(9) smo(0) asn1Modules(2) negotiationDefinitions(0) version1(1)}
DEFINITIONS ::= BEGIN

-- Additional syntax to support ACSE
```

```
-- CMIP functional units

-- CMIP user info defined in SMO
-- Abstract syntax name for a SMASE-A-Associate-Information.SMASEUserData is
--        Joint-iso-ccitt 9 0 1 1

SMASEUserData ::= SEQUENCE
              {smfuPackages SET OF FunctionalUnitPackage OPTIONAL,
               -- shall be present on request/indication if SMFU
               -- negotiation is proposed and on response/confirm if it
               -- is accepted, otherwise it shall be omitted
               reason Reason OPTIONAL,
               -- may only be present on the response/confirm.
               -- When SMFU negotiation fails
               --                       results in a reduction of proposed SMFUs
               --                    or the association request is rejected
               -- this parameter may carry a reason for this
               systemsManagementUserInformation GraphicString OPTIONAL
               -- a text bucket for implementations to use to distinguish
               -- between different implementation environments.
               -- not subject to conformance test.
               }

Reason ::= INTEGER
              {smfusNotSupported      (0),
               -- one or more of the proposed SMFUS is not supported
               smfuCombinationNotSupported  (1),
               -- the individual SMFUS are supported, but not in the
               -- proposed combination on a single association
               smfusRequiredNotAvailable    (2),
               -- one or more required SMFUs have been negotiated away
               smfuNegotiationRefused       (3)
               -- responder refuses to negotiate SMFUs without saying why
               }

FunctionalUnitPackage ::= SEQUENCE
              {functionUnitPackageId FunctionalUnitPackageId,
               managerRoleFunctionalUnit [0] IMPLICIT BIT STRING DEFAULT {},
               -- if not present implies role not supported for this functional unit package
               agentRoleFunctionalUnit [1] IMPLICIT BIT STRING DEFAULT {}
               -- if not present implies role not supported for this functional unit package
               }

FunctionalUnitPackageId ::= OBJECT IDENTIFIER
-- the values for the functionalUnitPackageId are
-- joint-iso-ccitt 9 2 X 1 where x is the defined by the standard (likely equal to the 10164-x)
-- so far, this is the case.

-- Request
-- (smfuPackages ((2.9.2.1.1,managerRoleFunctionalUnit 1111, agentRoleFunctionalUnit 1111),
--                 (2.9.2.2.1, managerRoleFunctionalUnit 1, agentRoleFunctionalUnit 1),
--                 (2.9.2.3.1, managerRoleFunctionalUnit 1, agentRoleFunctionalUnit 1),
--                 (2.9.2.4.1, managerRoleFunctionalUnit 1, agentRoleFunctionalUnit 1),
--                 (2.9.2.5.1, managerRoleFunctionalUnit 11, agentRoleFunctionalUnit 11),
--                 (2.9.2.6.1, managerRoleFunctionalUnit 00, agentRoleFunctionalUnit 00) -- --  NO LOGGING
--               )
--  )
-- response

-- (smfuPackages ((2.9.2.1.1,managerRoleFunctionalUnit (1111), agentRoleFunctionalUnit ()),
--                 (2.9.2.2.1, managerRoleFunctionalUnit (1), agentRoleFunctionalUnit ()),
--                 (2.9.2.3.1, managerRoleFunctionalUnit (1), agentRoleFunctionalUnit ()),
--                 (2.9.2.4.1, managerRoleFunctionalUnit (1), agentRoleFunctionalUnit ()),
--                 (2.9.2.5.1, managerRoleFunctionalUnit (11), agentRoleFunctionalUnit ()),
--                 (2.9.2.6.1, managerRoleFunctionalUnit (00), agentRoleFunctionalUnit ()), -- -- NO LOGGING
--               ),
--              reason 0
--  )
END

-- Rose defines its ASE id to be joint-iso-ccitt 4 3
-- BER is joint-iso-ccitt 1 1 for the transfer syntax. this will always be used.

CMIP-A-ASSOCIATE-Information {joint-iso-ccitt ms(9) cmip(1) modules(0)aAssociateUserInfo(1)}
DEFINITIONS ::= BEGIN
--EXPORTS everything

FunctionalUnits::= BIT STRING --+ VL-NAME = CMIS-Functional-Units
                             { multipleObjectSelection (0)
                               --+ ELEM-NAME = multiple-Object-Selection
                               --+ H-ELEM-NAME = "MP_T_FU_MULTIPLE_FUNCTIONAL_UNITS"
                               --+ H-ELEM-ID = 4
                               ,
                               filter (1)
                               --+ ELEM-NAME = filter
                               --+ H-ELEM-NAME = "MP_T_FU_FILTER"
                               --+ H-ELEM-ID = 2
                               ,
```

```
                                multipleReply (2)
                                --+ ELEM-NAME = multiple-Reply
                                --+ H-ELEM-NAME = "MP_T_FU_MULTIPLE_REPLY"
                                --+ H-ELEM-ID = 16
                                ,
                                extendedService (3)
                                --+ ELEM-NAME = extended-Service
                                --+ H-ELEM-NAME = "MP_T_FU_EXTENDED_SERVICE"
                                --+ H-ELEM-ID = 8
                                ,
                                cancelGet (4)
                                --+ ELEM-NAME = cancel-Get
                                --+ H-ELEM-NAME = "MP_T_FU_CANCEL_GET"
                                --+ H-ELEM-ID = 1
                                }

-- Functional unit i is supported if and only if bit i is one.
-- information carried in user-information parameter of A-ASSOCIATE

CMIPUserInfo::= SEQUENCE { protocolVersion [0] IMPLICIT ProtocolVersion DEFAULT { version1 },
                           functionalUnits [1] IMPLICIT FunctionalUnits DEFAULT {},
                           accessControl   [2] EXTERNAL OPTIONAL,
                           userInfo        [3] EXTERNAL OPTIONAL }

ProtocolVersion::= BIT STRING { version1 (0),
                                version2 (1) }

END



CMIP-A-ABORT-Information {joint-iso-ccitt ms(9) cmip(1) modules(0)aAbortUserInfo(2)}
DEFINITIONS ::= BEGIN

-- information carried in user-information parameter of A-ABORT

CMIPAbortInfo ::= SEQUENCE {   abortSource   [0] IMPLICIT CMIPAbortSource,
                               userInfo      [1] EXTERNAL OPTIONAL }

CMIPAbortSource ::= ENUMERATED { cmiseServiceUser     (0),
                                 cmiseServiceProvider (1) }

END


CMIP-1 {joint-iso-ccitt ms(9) cmip(1) modules(0) protocol(3)}

DEFINITIONS ::= BEGIN

-- The IMPORTS statement was removed to allow compilation without ROSE.asn
-- IMPORTS InvokeIDType, Operation, Error
-- FROM Remote-Operations-APDUs ;
-- EXPORTS everything

-- Directory Service definitions
   IMPORTS RDNSequence, DistinguishedName
    FROM InformationFramework {joint-iso-ccitt ds(5) modules(1) informationFramework(1)};

-- EXPORTS DistinguishedName, RDN;

-- added to allow compilation of the CMP file without ROSE
-- and without the rose macros.
InvokeIDType ::= INTEGER
Operation    ::= INTEGER
Error        ::= INTEGER


-- ADDED to allow extern information for access control
ExternDefault ::= [UNIVERSAL 8] IMPLICIT SEQUENCE {
               direct-reference OBJECT IDENTIFIER OPTIONAL,
               indirect-reference INTEGER OPTIONAL,
               encoding CHOICE {
                  single-ASN1-type    [0] ANY,
                  octet-aligned       [1] IMPLICIT OCTET STRING,
                  arbitrary           [2] IMPLICIT BIT STRING

                  }
               }

-- Added to allow the missingAttributeValue error syntax to be resolved
-- The parameter template says SET OF AttributeId - not a very friendly type.
AttributeIds ::= SET OF AttributeId

ROSEapdus ::= CHOICE {
     roiv-apdu [1] IMPLICIT ROIVapdu,
     rors-apdu [2] IMPLICIT RORSapdu,
     roer-apdu [3] IMPLICIT ROERapdu,
     rorj-apdu [4] IMPLICIT RORJapdu
```

```
     }


ROIVapdu ::= [1] IMPLICIT SEQUENCE
   {invokeID        InvokeIDType,
    linked-ID   [0] IMPLICIT InvokeIDType OPTIONAL,
    operation-value Operation,
    argument        ANY DEFINED BY operation-value --% ANY_TABLE_REF(Operations) %-- OPTIONAL
    }

--% Operations ANY_TABLE ::=
--% {
--%     m-EventReport            EventReportArgument,
--%     m-EventReport-Confirmed  EventReportArgument,
--%     m-Linked-Reply           LinkedReplyArgument,
--%     m-Get                    GetArgument,
--%     m-Set                    SetArgument,
--%     m-Set-Confirmed          SetArgument,
--%     m-Action                 ActionArgument,
--%     m-Action-Confirmed       ActionArgument,
--%     m-Create                 CreateArgument,
--%     m-Delete                 DeleteArgument,
--%     m-CancelGet              InvokeIDType
--% }

m-EventReport            Operation ::= 0
m-EventReport-Confirmed  Operation ::= 1
m-Linked-Reply           Operation ::= 2
m-Get                    Operation ::= 3
m-Set                    Operation ::= 4
m-Set-Confirmed          Operation ::= 5
m-Action                 Operation ::= 6
m-Action-Confirmed       Operation ::= 7
m-Create                 Operation ::= 8
m-Delete                 Operation ::= 9
m-CancelGet              Operation ::= 10


RORSapdu ::= [2] IMPLICIT SEQUENCE
   {invokeID      InvokeIDType,
    resultOption  SEQUENCE
      {operation-value    Operation,
       result             ANY DEFINED BY operation-value --% ANY_TABLE_REF(Results)
      } OPTIONAL
   }
--  Note that m-CancelGet is not included in the list. This message does not
--    have an associated parameter and should only be responed to with an invokeID
--% Results ANY_TABLE ::=
--% {
--%     m-Action-Confirmed       ActionResult,
--%     m-Create                 CreateResult,
--%     m-Delete                 DeleteResult,
--%     m-EventReport-Confirmed  EventReportResult,
--%     m-Get                    GetResult,
--%     m-Set-Confirmed          SetResult
--% }

ROERapdu ::= [3] IMPLICIT SEQUENCE
   {invokeID     InvokeIDType,
    error-value  Error,
    parameter    ANY DEFINED BY error-value --% ANY_TABLE_REF(Errors) %-- OPTIONAL
    }
--  Note that the errors accessDenied, mistypedOperation and operationCancelled
--   are not included in the following list. These errors do not have information
--   associated with them so the 'parameter' field should never be present.
--% Errors ANY_TABLE ::=
--% {
--%     classInstanceConflict    BaseManagedObjectId,
--%     complexityLimitation     ComplexityLimitation,
--%     duplicateManagedObjectInstance  ObjectInstance,
--%     getListError             GetListError,
--%     invalidArgumentValue     InvalidArgumentValue,
--%     invalidAttributeValue    Attribute,
--%     invalidFilter            CMISFilter,
--%     invalidObjectInstance    ObjectInstance,
--%     invalidScope             Scope,
--%     missingAttributeValue    AttributeIds,
--%     noSuchAction             NoSuchAction,
--%     noSuchArgument           NoSuchArgument,
--%     noSuchAttribute          AttributeId,
--%     noSuchEventType          NoSuchEventType,
--%     noSuchInvokeId           InvokeIDType,
--%     noSuchObjectClass        ObjectClass,
--%     noSuchObjectInstance     ObjectInstance,
--%     noSuchReferenceObject    ObjectInstance,
--%     processingFailure        ProcessingFailure,
--%     setListError             SetListError,
--%     syncNotSupported         CMISSync
```

```
--% }

accessDenied                   Error ::= 2
classInstanceConflict          Error ::= 19
complexityLimitation           Error ::= 20
duplicateManagedObjectInstance Error ::= 11
getListError                   Error ::= 7
invalidArgumentValue           Error ::= 15
invalidAttributeValue          Error ::= 6
invalidFilter                  Error ::= 4
invalidObjectInstance          Error ::= 17
invalidOperation               Error ::= 24
invalidScope                   Error ::= 16
missingAttributeValue          Error ::= 18
mistypedOperation              Error ::= 21
noSuchAction                   Error ::= 9
noSuchArgument                 Error ::= 14
noSuchAttribute                Error ::= 5
noSuchEventType                Error ::= 13
noSuchInvokeId                 Error ::= 22
noSuchObjectClass              Error ::= 0
noSuchObjectInstance           Error ::= 1
noSuchReferenceObject          Error ::= 12
operationCancelled             Error ::= 23
processingFailure              Error ::= 10
setListError                   Error ::= 8
syncNotSupported               Error ::= 3

-- Labels have been added to the problem CHOICE to allow it to be correctly processed
RORJapdu ::= [4] IMPLICIT SEQUENCE
   {invokeID   CHOICE{InvokeIDType,NULL},
    problem    CHOICE
      {generalProblem      [0] IMPLICIT GeneralProblem,
       invokeProblem       [1] IMPLICIT InvokeProblem,
       returnResultProblem [2] IMPLICIT ReturnResultProblem,
       returnErrorProblem  [3] IMPLICIT ReturnErrorProblem
      }
   }


-- The following problems are detected by ROSE-providers:

GeneralProblem ::= INTEGER
   {
    unrecognisedAPDU(0),
    mistypedAPDU(1),
    badlyStructuredAPDU(2)
    }

-- The following problems are detected by ROSE-users:

InvokeProblem ::= INTEGER
   {
    duplicateInvocation(0),
    unrecognisedOperation(1),
    mistypedArgument(2),
    resourceLimitation(3),
    initiatorReleasing(4),
    unrecognizedLinkedID(5),
    linkedResponseUnexpected(6),
    unexpectedChildOperation(7)
    }

ReturnResultProblem ::= INTEGER
   {
    unrecognisedInvocation(0),
    resultResponseUnexpected(1),
    mistypedResponse(2)
    }

ReturnErrorProblem ::= INTEGER
   {
    unrecognisedInvocation(0),
    errorResponseUnexpected(1),
    unrecognisedError(2),
    unexpectedError(3),
    mistypedParameter(4)
    }

AccessControl::= --+ CL-NAME = Access-Control
                 --+ CL-TYPE = 5
                 --+ H-CL-NAME = "OMP_O_MP_C_ACCESS_CONTROL"
                 --+ H-CL-ID = 1001
                 ExternDefault    -- EXTERNAL in 9596


ActionArgument::= --+ SUPER-CLASS = Action-Argument
                  --+ CL-NAME = CMIS-Action-Argument
```

```
                        --+ CL-TYPE = 3
                        --+ H-CL-NAME = "OMP_O_MP_C_CMIS_ACTION_ARGUMENT"
                        --+ H-CL-ID = 2012
                        SEQUENCE { baseManagedObjectClass       ObjectClass
                                --+ ATTR-NAME = base-Managed-Object-Class
                                --+ H-ATTR-NAME = "MP_BASE_MANAGED_OBJECT_CLASS"
                                --+ H-ATTR-ID = 2023
                                --+ ATTR-SYNTAX = 127 "Object-Class"
                                --+ VALUE-NUMBER = 1
                            ,
                            baseManagedObjectInstance    ObjectInstance
                                --+ ATTR-NAME = base-Managed-Object-Instance
                                --+ H-ATTR-NAME = "MP_BASE_MANAGED_OBJECT_INSTANCE"
                                --+ H-ATTR-ID = 2024
                                --+ ATTR-SYNTAX = 127 "Object-Instance"
                                --+ VALUE-NUMBER = 1
                            ,
                            accessControl [5] AccessControl OPTIONAL
                                --+ ATTR-NAME = access-Control
                                --+ H-ATTR-NAME = "MP_ACCESS_CONTROL"
                                --+ H-ATTR-ID = 1001
                                --+ ATTR-SYNTAX = 127 "Access-Control"
                                --+ VALUE-NUMBER = 0
                            ,
                            synchronization [6] IMPLICIT CMISSync DEFAULT bestEffort
                                --+ ATTR-NAME = synchronization
                                --+ H-ATTR-NAME = "MP_SYNCHRONIZATION"
                                --+ H-ATTR-ID = 2080
                                --+ ATTR-SYNTAX = 10 "CMIS-Sync"
                                --+ VALUE-NUMBER = 0
                            ,
                            scope [7] Scope DEFAULT basicScope : baseObject
                                --+ ATTR-NAME = scope
                                --+ H-ATTR-NAME = "MP_SCOPE"
                                --+ H-ATTR-ID = 2070
                                --+ ATTR-SYNTAX = 127 "Scope"
                                --+ VALUE-NUMBER = 0
                            ,
                            filter CMISFilter DEFAULT and:{}
                                --+ ATTR-NAME = filter
                                --+ H-ATTR-NAME = "MP_FILTER"
                                --+ H-ATTR-ID = 2043
                                --+ ATTR-SYNTAX = 127 "CMIS-Filter"
                                --+ VALUE-NUMBER = 0
                            ,
                            actionInfo [12] IMPLICIT ActionInfo
                                --+ ATTR-NAME = action-Info
                                --+ H-ATTR-NAME = "MP_ACTION_INFO"
                                --+ H-ATTR-ID = 2005
                                --+ ATTR-SYNTAX = 127 "Action-Info"
                                --+ VALUE-NUMBER = 1
                            }


ActionError::= --+ CL-NAME = Action-Error
               --+ CL-TYPE = 3
               --+ H-CL-NAME = "OMP_O_MP_C_ACTION_ERROR"
               --+ H-CL-ID = 2001
               SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                           --+ ATTR-NAME = managed-Object-Class
                           --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                           --+ H-ATTR-ID = 2057
                           --+ ATTR-SYNTAX = 127 "Object-Class"
                           --+ VALUE-NUMBER = 0
                        ,
                        managedObjectInstance ObjectInstance OPTIONAL
                            --+ ATTR-NAME = managed-Object-Instance
                            --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                            --+ H-ATTR-ID = 2058
                            --+ ATTR-SYNTAX = 127 "Object-Instance"
                            --+ VALUE-NUMBER = 0
                        ,
                        currentTime [5] IMPLICIT GeneralizedTime OPTIONAL
                            --+ ATTR-NAME = current-Time
                            --+ H-ATTR-NAME = "MP_CURRENT_TIME"
                            --+ H-ATTR-ID = 2027
                            --+ ATTR-SYNTAX = 24
                            -- Time
                            --+ VALUE-NUMBER = 0
                        ,
                        actionErrorInfo [6] ActionErrorInfo
                            --+ ATTR-NAME = action-Error-Info
                            --+ H-ATTR-NAME = "MP_ACTION_ERROR_INFO"
                            --+ H-ATTR-ID = 2003
                            --+ ATTR-SYNTAX = 127 "Action-Error-Info"
                            --+ VALUE-NUMBER = 1
                        }
```

```
ActionErrorInfo::= --+ CL-NAME = Action-Error-Info
                   --+ CL-TYPE = 3
                   --+ H-CL-NAME = "OMP_O_MP_C_ACTION_ERROR_INFO"
                   --+ H-CL-ID = 2002
                   SEQUENCE { errorStatus ENUMERATED --+ VL-NAME = Error-Status
                                                     { accessDenied (2)
                                                       --+ ELEM-NAME = access-Denied
                                                       --+ H-ELEM-NAME = "MP_E_ACCESS_DENIED"
                                                       --+ H-ELEM-ID = 2
                                                       ,
                                                       noSuchAction (9)
                                                       --+ ELEM-NAME = no-Such-Action
                                                       --+ H-ELEM-NAME = "MP_E_NO_SUCH_ACTION"
                                                       --+ H-ELEM-ID = 9
                                                       ,
                                                       noSuchArgument (14)
                                                       --+ ELEM-NAME = no-Such-Argument
                                                       --+ H-ELEM-NAME = "MP_E_NO_SUCH_ARGUMENT"
                                                       --+ H-ELEM-ID = 14
                                                       ,
                                                       invalidArgumentValue (15)
                                                       --+ ELEM-NAME = invalid-Argument-Value
                                                       --+ H-ELEM-NAME = "MP_E_INVALID_ARGUMENT_VALUE"
                                                       --+ H-ELEM-ID = 15
                                                       }
                                --+ ATTR-NAME = error-Status
                                --+ H-ATTR-NAME = "MP_ERROR_STATUS"
                                --+ H-ATTR-ID = 2035
                                --+ ATTR-SYNTAX = 10 "Error-Status"
                                --+ VALUE-NUMBER = 1
                              ,
                              errorInfo CHOICE { actionType ActionTypeId
                                                   --+ ATTR-NAME = action-Type
                                                   --+ H-ATTR-NAME = "MP_ACTION_TYPE"
                                                   --+ H-ATTR-ID = 2010
                                                   --+ ATTR-SYNTAX = 127 "Action-Type-Id"
                                                   --+ VALUE-NUMBER = 0
                                                 ,
                                                 actionArgument [0] NoSuchArgument
                                                   --+ ATTR-NAME = action-Argument
                                                   --+ H-ATTR-NAME = "MP_ACTION_ARGUMENT"
                                                   --+ H-ATTR-ID = 2001
                                                   --+ ATTR-SYNTAX = 127 "No-Such-Argument"
                                                   --+ VALUE-NUMBER = 0
                                                 ,
                                                 argumentValue [1] InvalidArgumentValue
                                                   --+ ATTR-NAME = argument-Value
                                                   --+ H-ATTR-NAME = "MP_ARGUMENT_VALUE"
                                                   --+ H-ATTR-ID = 2014
                                                   --+ ATTR-SYNTAX = 127 "Invalid-Argument-Value"
                                                   --+ VALUE-NUMBER = 0
                                                 }
                                 --+ ATTR-NAME = error-Info
                                 --+ H-ATTR-NAME = "MP_ERROR_INFO"
                                 --+ H-ATTR-ID = 2034
                                 --+ ATTR-SYNTAX = 127 "Error-Info"
                                 --+ VALUE-NUMBER = 1
                                 --+ CL-NAME = Error-Info
                                 --+ CL-TYPE = 0
                                 --+ H-CL-NAME = "OMP_O_MP_C_ERROR_INFO"
                                 --+ H-CL-ID = 2033
                              }


ActionInfo::= --+ CL-NAME = Action-Info
              --+ CL-TYPE = 3
              --+ H-CL-NAME = "OMP_O_MP_C_ACTION_INFO"
              --+ H-CL-ID = 2003
              SEQUENCE { actionType ActionTypeId
                             --+ ATTR-NAME = action-Type
                             --+ H-ATTR-NAME = "MP_ACTION_TYPE"
                             --+ H-ATTR-ID = 2010
                             --+ ATTR-SYNTAX = 127 "Action-Type-Id"
                             --+ VALUE-NUMBER = 1
                         ,
                         actionInfoArg [4] ANY DEFINED BY actionType
                             --% ANY_TABLE_REF(ActionInfoTableMod.ActionInfoTypes) %-- OPTIONAL
                             --+ ATTR-NAME = action-Info-Arg
                             --+ H-ATTR-NAME = "MP_ACTION_INFO_ARG"
                             --+ H-ATTR-ID = 2006
                             --+ ATTR-SYNTAX = 8
                             --+ VALUE-NUMBER = 0
                         }


ActionReply::= --+ CL-NAME = Action-Reply
               --+ CL-TYPE = 3
```

```
                --+ H-CL-NAME = "OMP_O_MP_C_ACTION_REPLY"
                --+ H-CL-ID = 2004
                SEQUENCE { actionType ActionTypeId
                               --+ ATTR-NAME = action-Type
                               --+ H-ATTR-NAME = "MP_ACTION_TYPE"
                               --+ H-ATTR-ID = 2010
                               --+ ATTR-SYNTAX = 127 "Action-Type-Id"
                               --+ VALUE-NUMBER = 1
                          ,
                          actionReplyInfo [4] ANY DEFINED BY actionType
                              --% ANY_TABLE_REF(ActionReplyTableMod.ActionReplyTypes)
                              --+ ATTR-NAME = action-Reply-Info
                              --+ H-ATTR-NAME = "MP_ACTION_REPLY_INFO"
                              --+ H-ATTR-ID = 2008
                              --+ ATTR-SYNTAX = 8
                              --+ VALUE-NUMBER = 1
                          }

ActionResult::= --+ SUPER-CLASS = Action-Result
                 --+ CL-NAME = CMIS-Action-Result
                 --+ CL-TYPE = 3
                 --+ H-CL-NAME = "OMP_O_MP_C_CMIS_ACTION_RESULT"
                 --+ H-CL-ID = 2013
                 SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                                --+ ATTR-NAME = managed-Object-Class
                                --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                                --+ H-ATTR-ID = 2057
                                --+ ATTR-SYNTAX = 127 "Object-Class"
                                --+ VALUE-NUMBER = 0
                           ,
                           managedObjectInstance ObjectInstance OPTIONAL
                                --+ ATTR-NAME = managed-Object-Instance
                                --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                                --+ H-ATTR-ID = 2058
                                --+ ATTR-SYNTAX = 127 "Object-Instance"
                                --+ VALUE-NUMBER = 0
                           ,
                           currentTime [5] IMPLICIT GeneralizedTime OPTIONAL
                                --+ ATTR-NAME = current-Time
                                --+ H-ATTR-NAME = "MP_CURRENT_TIME"
                                --+ H-ATTR-ID = 2027
                                --+ ATTR-SYNTAX = 24
                                --+ VALUE-NUMBER = 0
                           ,
                           actionReply [6] IMPLICIT ActionReply OPTIONAL
                                --+ ATTR-NAME = action-Reply
                                --+ H-ATTR-NAME = "MP_ACTION_REPLY"
                                --+ H-ATTR-ID = 2007
                                --+ ATTR-SYNTAX = 127 "Action-Reply"
                                --+ VALUE-NUMBER = 0
                           }

-- This has been adapted to align with the comments below.
-- The CHOICE has been eliminated to simplify processing
-- and the use of the API.
ActionTypeId::= [2] IMPLICIT OBJECT IDENTIFIER
-- This [Recommendation | part of ISO/IEC 9596] does not allocate any values for
-- localForm.
-- Where this alternative is used, the permissible values for the integers
-- and their meanings shall be defined as part of the application context
-- in which they are used.


Attribute::= --+ CL-NAME = Attribute
             --+ CL-TYPE = 3
             --+ H-CL-NAME = "OMP_O_MP_C_ATTRIBUTE"
             --+ H-CL-ID = 2006
             SEQUENCE { attributeId AttributeId
                            --+ ATTR-NAME = attribute-Id
                            --+ H-ATTR-NAME = "MP_ATTRIBUTE_ID"
                            --+ H-ATTR-ID = 2017
                            --+ ATTR-SYNTAX = 127 "Attribute-Id"
                            --+ VALUE-NUMBER = 1
                       ,
                       attributeValue ANY DEFINED BY attributeId
                           --% ANY_TABLE_REF(AttributeTableMod.AttributeTypes)
                           --+ ATTR-NAME = attribute-Value
                           --+ H-ATTR-NAME = "MP_ATTRIBUTE_VALUE"
                           --+ H-ATTR-ID = 2022
                           --+ ATTR-SYNTAX = 8
                           --+ VALUE-NUMBER = 1
                       }

AttributeError::= --+ CL-NAME = Attribute-Error
                  --+ CL-TYPE = 3
                  --+ H-CL-NAME = "OMP_O_MP_C_ATTRIBUTE_ERROR"
                  --+ H-CL-ID = 2007
                  SEQUENCE { errorStatus ENUMERATED --+ VL-NAME = Error-Status
```

```
                                                { accessDenied (2),
                                                  noSuchAttribute (5)
                                                  --+ ELEM-NAME = no-Such-Attribute
                                                  --+ H-ELEM-NAME = "MP_E_NO_SUCH_ATTRIBUTE"
                                                  --+ H-ELEM-ID = 5
                                                  ,
                                                  invalidAttributeValue (6)
                                                  --+ ELEM-NAME = invalid-Attribute-Value
                                                  --+ H-ELEM-NAME = "MP_E_INVALID_ATTRIBUTE_VALUE"
                                                  --+ H-ELEM-ID = 6
                                                  ,
                                                  invalidOperation (24)
                                                  --+ ELEM-NAME = invalid-Operation
                                                  --+ H-ELEM-NAME = "MP_E_INVALID_OPERATION"
                                                  --+ H-ELEM-ID = 24
                                                  ,
                                                  invalidOperator (25)
                                                  --+ ELEM-NAME = invalid-Operator
                                                  --+ H-ELEM-NAME = "MP_E_INVALID_OPERATOR"
                                                  --+ H-ELEM-ID = 25
                                                  }
                                        --+ ATTR-NAME = error-Status
                                        --+ H-ATTR-NAME = "MP_ERROR_STATUS"
                                        --+ H-ATTR-ID = 2035
                                        --+ ATTR-SYNTAX = 10 "Error-Status"
                                        --+ VALUE-NUMBER = 1
                                    ,
                                    modifyOperator [2] IMPLICIT ModifyOperator OPTIONAL
                                        --+ ATTR-NAME = modify-Operator
                                        --+ H-ATTR-NAME = "MP_MODIFY_OPERATOR"
                                        --+ H-ATTR-ID = 2060
                                        --+ ATTR-SYNTAX = 10 "Modify-Operator"
                                        --+ VALUE-NUMBER = 0
                                    ,
                                    attributeId AttributeId
                                        --+ ATTR-NAME = attribute-Id
                                        --+ H-ATTR-NAME = "MP_ATTRIBUTE_ID"
                                        --+ H-ATTR-ID = 2017
                                        --+ ATTR-SYNTAX = 127 "Attribute-Id"
                                        --+ VALUE-NUMBER = 1
                                    ,
                                    attributeValue ANY DEFINED BY attributeId
                                        --% ANY_TABLE_REF(AttributeTableMod.AttributeTypes) %-- OPTIONAL
                                        --+ ATTR-NAME = attribute-Value
                                        --+ H-ATTR-NAME = "MP_ATTRIBUTE_VALUE"
                                        --+ H-ATTR-ID = 2022
                                        --+ ATTR-SYNTAX = 8
                                        --+ VALUE-NUMBER = 0
                                    }

-- This has been adapted to align with the comments below.
-- The CHOICE has been eliminated to simplify processing
-- and the use of the API.
AttributeId ::= [0] IMPLICIT OBJECT IDENTIFIER
-- This [Recommendation | part of ISO/IEC 9596] does not allocate any values for
-- localForm.
-- Where this alternative is used, the permissible values for the integers
-- and their meanings shall be defined as part of the application context
-- in which they are used.

AttributeIdError::= --+ CL-NAME = Attribute-Id-Error
                    --+ CL-TYPE = 3
                    --+ H-CL-NAME = "OMP_O_MP_C_ATTRIBUTE_ID_ERROR"
                    --+ H-CL-ID = 2009
                    SEQUENCE { errorStatus ENUMERATED { accessDenied (2),
                                                        noSuchAttribute (5) }
                                    --+ ATTR-NAME = error-Status
                                    --+ H-ATTR-NAME = "MP_ERROR_STATUS"
                                    --+ H-ATTR-ID = 2035
                                    --+ ATTR-SYNTAX = 10 "Error-Status"
                                    --+ VALUE-NUMBER = 1
                               ,
                               attributeId AttributeId
                                    --+ ATTR-NAME = attribute-Id
                                    --+ H-ATTR-NAME = "MP_ATTRIBUTE_ID"
                                    --+ H-ATTR-ID = 2017
                                    --+ ATTR-SYNTAX = 127 "Attribute-Id"
                                    --+ VALUE-NUMBER = 1
                               }

BaseManagedObjectId::= --+ CL-NAME = Base-Managed-Object-Id
                       --+ CL-TYPE = 3
                       --+ H-CL-NAME = "OMP_O_MP_C_BASE_MANAGED_OBJETC_ID"
                       --+ H-CL-ID = 2011
                       SEQUENCE { baseManagedObjectClass ObjectClass
                                       --+ ATTR-NAME = base-Managed-Object-Class
                                       --+ H-ATTR-NAME = "MP_BASE_MANAGED_OBJECT_CLASS"
                                       --+ H-ATTR-ID = 2023
```

```
                                    --+ ATTR-SYNTAX = 127 "Object-Class"
                                    --+ VALUE-NUMBER = 1
                                ,
                                baseManagedObjectInstance ObjectInstance
                                    --+ ATTR-NAME = base-Managed-Object-Instance
                                    --+ H-ATTR-NAME = "MP_BASE_MANAGED_OBJECT_INSTANCE"
                                    --+ H-ATTR-ID = 2024
                                    --+ ATTR-SYNTAX = 127 "Object-Instance"
                                    --+ VALUE-NUMBER = 1
                                }


CMISFilter::= --+ CL-NAME = CMIS-Filter
              --+ CL-TYPE = 0
              --+ H-CL-NAME = "OMP_O_MP_C_CMIS_FILTER"
              --+ H-CL-ID = 2021
              CHOICE { item [8] FilterItem
                           --+ ATTR-NAME = item
                           --+ H-ATTR-NAME = "MP_ITEM"
                           --+ H-ATTR-ID = 2053
                           --+ ATTR-SYNTAX = 127 "Filter-Item"
                           --+ VALUE-NUMBER = 0
                     ,
                     and [9] IMPLICIT SET OF CMISFilter
                           --+ ATTR-NAME = and
                           --+ H-ATTR-NAME = "MP_AND"
                           --+ H-ATTR-ID = 2012
                           --+ ATTR-SYNTAX = 127 "CMIS-Filter"
                           --+ VALUE-NUMBER = 2
                     ,
                     or [10] IMPLICIT SET OF CMISFilter
                           --+ ATTR-NAME = or
                           --+ H-ATTR-NAME = "MP_OR"
                           --+ H-ATTR-ID = 2065
                           --+ ATTR-SYNTAX = 127 "CMIS-Filter"
                           --+ VALUE-NUMBER = 2
                     ,
                     not [11] CMISFilter
                           --+ ATTR-NAME = not
                           --+ H-ATTR-NAME = "MP_NOT"
                           --+ H-ATTR-ID = 2064
                           --+ ATTR-SYNTAX = 127 "CMIS-Filter"
                           --+ VALUE-NUMBER = 0
                     }

CMISSync::= ENUMERATED --+ VL-NAME = CMIS-Sync
                       { bestEffort (0)
                         --+ ELEM-NAME = best-effort
                         --+ H-ELEM-NAME = "MP_T_BEST_EFFORT"
                         --+ H-ELEM-ID = 0
                         ,
                         atomic (1)
                         --+ ELEM-NAME = atomic
                         --+ H-ELEM-NAME = "MP_T_ATOMIC"
                         --+ H-ELEM-ID = 1
                         }


ComplexityLimitation::= --+ CL-NAME = Complexity-Limitation
                        --+ CL-TYPE = 1
                        --+ H-CL-NAME = "OMP_O_MP_C_COMPLEXITY_LIMITATION"
                        --+ H-CL-ID = 2030
                        SET { scope [0] Scope OPTIONAL
                                  --+ ATTR-NAME = scope
                                  --+ H-ATTR-NAME = "MP_SCOPE"
                                  --+ H-ATTR-ID = 2070
                                  --+ ATTR-SYNTAX = 127 "Scope"
                                  --+ VALUE-NUMBER = 0
                              ,
                              filter [1] CMISFilter OPTIONAL
                                  --+ ATTR-NAME = filter
                                  --+ H-ATTR-NAME = "MP_FILTER"
                                  --+ H-ATTR-ID = 2043
                                  --+ ATTR-SYNTAX = 127 "CMIS-Filter"
                                  --+ VALUE-NUMBER = 0
                              ,
                              sync [2] CMISSync OPTIONAL
                                  --+ ATTR-NAME = synchronization
                                  --+ H-ATTR-NAME = "MP_SYNCHRONIZATION"
                                  --+ H-ATTR-ID = 2080
                                  --+ ATTR-SYNTAX = 10 "CMIS-Sync"
                                  --+ VALUE-NUMBER = 0
                              }

CreateArgument::= --+ SUPER-CLASS = Create-Argument
                  --+ CL-NAME = CMIS-Create-Argument
                  --+ CL-TYPE = 3
                  --+ H-CL-NAME = "OMP_O_MP_C_CMIS_CREATE_ARGUMENT"
```

```
                        --+ H-CL-ID = 2015
                        SEQUENCE { managedObjectClass ObjectClass
                                        --+ ATTR-NAME = managed-Object-Class
                                        --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                                        --+ H-ATTR-ID = 2057
                                        --+ ATTR-SYNTAX = 127 "Object-Class"
                                        --+ VALUE-NUMBER = 1
                                   ,
                                   CHOICE { managedObjectInstance ObjectInstance
                                                --+ ATTR-NAME = managed-Object-Instance
                                                --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                                                --+ H-ATTR-ID = 2058
                                                --+ ATTR-SYNTAX = 127 "Object-Instance"
                                                --+ VALUE-NUMBER = 0
                                           ,
                                           superiorObjectInstance [8] ObjectInstance
                                               --+ ATTR-NAME = superior-Object-Instance
                                               --+ H-ATTR-NAME = "MP_SUPERIOR_OBJECT_INSTANCE"
                                               --+ H-ATTR-ID = 2078
                                               --+ ATTR-SYNTAX = 127 "Object-Instance"
                                               --+ VALUE-NUMBER = 0
                                           } OPTIONAL
                                       --+ ATTR-NAME = create-Object-Instance
                                       --+ H-ATTR-NAME = "MP_CREATE_OBJECT_INSTANCE"
                                       --+ H-ATTR-ID = 2026
                                       --+ ATTR-SYNTAX = 127 "Create-Object-Instance"
                                       --+ VALUE-NUMBER = 0
                                       --+ CL-NAME = Create-Object-Instance
                                       --+ CL-TYPE = 0
                                       --+ H-CL-NAME = "OMP_O_MP_C_CREATE_OBJECT_INSTANCE"
                                       --+ H-CL-ID = 2031
                                   ,
                                   accessControl [5] AccessControl OPTIONAL
                                       --+ ATTR-NAME = access-Control
                                       --+ H-ATTR-NAME = "MP_ACCESS_CONTROL"
                                       --+ H-ATTR-ID = 1001
                                       --+ ATTR-SYNTAX = 127 "Access-Control"
                                       --+ VALUE-NUMBER = 0
                                   ,
                                   referenceObjectInstance [6] ObjectInstance OPTIONAL
                                       --+ ATTR-NAME = reference-Object-Instance
                                       --+ H-ATTR-NAME = "MP_REFERENCE_OBJECT_INSTANCE"
                                       --+ H-ATTR-ID = 2068
                                       --+ ATTR-SYNTAX = 127 "Object-Instance"
                                       --+ VALUE-NUMBER = 0
                                   ,
                                   attributeList [7] IMPLICIT SET OF Attribute OPTIONAL
                                       --+ ATTR-NAME = attribute-List
                                       --+ H-ATTR-NAME = "MP_ATTRIBUTE_LIST"
                                       --+ H-ATTR-ID = 2021
                                       --+ ATTR-SYNTAX = 127 "Attribute"
                                       --+ VALUE-NUMBER = 2
                                   }


CreateResult::= --+ SUPER-CLASS = Create-Result
                --+ CL-NAME = CMIS-Create-Result
                --+ CL-TYPE = 3
                --+ H-CL-NAME = "OMP_O_MP_C_CMIS_CREATE_RESULT"
                --+ H-CL-ID = 2016
                SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                               --+ ATTR-NAME = managed-Object-Class
                               --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                               --+ H-ATTR-ID = 2057
                               --+ ATTR-SYNTAX = 127 "Object-Class"
                               --+ VALUE-NUMBER = 0
                           ,
                           managedObjectInstance ObjectInstance OPTIONAL
                               --+ ATTR-NAME = managed-Object-Instance
                               --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                               --+ H-ATTR-ID = 2058
                               --+ ATTR-SYNTAX = 127 "Object-Instance"
                               --+ VALUE-NUMBER = 0
                           ,
                           -- shall be returned if omitted from CreateArgument
                           currentTime [5] IMPLICIT GeneralizedTime OPTIONAL
                               --+ ATTR-NAME = current-Time
                               --+ H-ATTR-NAME = "MP_CURRENT_TIME"
                               --+ H-ATTR-ID = 2027
                               --+ ATTR-SYNTAX = 24
                               --+ VALUE-NUMBER = 0
                           ,
                           attributeList [6] IMPLICIT SET OF Attribute OPTIONAL
                               --+ ATTR-NAME = attribute-List
                               --+ H-ATTR-NAME = "MP_ATTRIBUTE_LIST"
                               --+ H-ATTR-ID = 2021
                               --+ ATTR-SYNTAX = 127 "Attribute"
                               --+ VALUE-NUMBER = 2
```

```
                              }


DeleteArgument::= --+ SUPER-CLASS = Delete-Argument
                  --+ CL-NAME = CMIS-Delete-Argument
                  --+ CL-TYPE = 3
                  --+ H-CL-NAME = "OMP_O_MP_C_CMIS_DELETE_ARGUMENT"
                  --+ H-CL-ID = 2017
                  SEQUENCE { baseManagedObjectClass       ObjectClass
                                  --+ ATTR-NAME = base-Managed-Object-Class
                                  --+ H-ATTR-NAME = "MP_BASE_MANAGED_OBJECT_CLASS"
                                  --+ H-ATTR-ID = 2023
                                  --+ ATTR-SYNTAX = 127 "Object-Class"
                                  --+ VALUE-NUMBER = 1
                           ,
                           baseManagedObjectInstance    ObjectInstance
                               --+ ATTR-NAME = base-Managed-Object-Instance
                               --+ H-ATTR-NAME = "MP_BASE_MANAGED_OBJECT_INSTANCE"
                               --+ H-ATTR-ID = 2024
                               --+ ATTR-SYNTAX = 127 "Object-Instance"
                               --+ VALUE-NUMBER = 1
                           ,
                           accessControl [5] AccessControl OPTIONAL
                               --+ ATTR-NAME = access-Control
                               --+ H-ATTR-NAME = "MP_ACCESS_CONTROL"
                               --+ H-ATTR-ID = 1001
                               --+ ATTR-SYNTAX = 127 "Access-Control"
                               --+ VALUE-NUMBER = 0
                           ,
                           synchronization [6] IMPLICIT CMISSync DEFAULT bestEffort
                               --+ ATTR-NAME = synchronization
                               --+ H-ATTR-NAME = "MP_SYNCHRONIZATION"
                               --+ H-ATTR-ID = 2080
                               --+ ATTR-SYNTAX = 10 "CMIS-Sync"
                               --+ VALUE-NUMBER = 0
                           ,
                           scope [7] Scope DEFAULT basicScope : baseObject
                               --+ ATTR-NAME = scope
                               --+ H-ATTR-NAME = "MP_SCOPE"
                               --+ H-ATTR-ID = 2070
                               --+ ATTR-SYNTAX = 127 "Scope"
                               --+ VALUE-NUMBER = 0
                           ,
                           filter CMISFilter DEFAULT and:{}
                               --+ ATTR-NAME = filter
                               --+ H-ATTR-NAME = "MP_FILTER"
                               --+ H-ATTR-ID = 2043
                               --+ ATTR-SYNTAX = 127 "CMIS-Filter"
                               --+ VALUE-NUMBER = 0
                           }


DeleteError::= --+ CL-NAME = Delete-Error
               --+ CL-TYPE = 3
               --+ H-CL-NAME = "OMP_O_MP_C_DELETE_ERROR"
               --+ H-CL-ID = 2032
               SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                             --+ ATTR-NAME = managed-Object-Class
                             --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                             --+ H-ATTR-ID = 2057
                             --+ ATTR-SYNTAX = 127 "Object-Class"
                             --+ VALUE-NUMBER = 0
                          ,
                          managedObjectInstance ObjectInstance OPTIONAL
                              --+ ATTR-NAME = managed-Object-Instance
                              --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                              --+ H-ATTR-ID = 2058
                              --+ ATTR-SYNTAX = 127 "Object-Instance"
                              --+ VALUE-NUMBER = 0
                          ,
                          currentTime [5] IMPLICIT GeneralizedTime OPTIONAL
                              --+ ATTR-NAME = current-Time
                              --+ H-ATTR-NAME = "MP_CURRENT_TIME"
                              --+ H-ATTR-ID = 2027
                              --+ ATTR-SYNTAX = 24
                              --+ VALUE-NUMBER = 0
                          ,
                          deleteErrorInfo [6] ENUMERATED --+ VL-NAME = Delete-Error-Info
                                                         { accessDenied (2)
                                                           --+ ELEM-NAME = access-Denied
                                                           --+ H-ELEM-NAME = "MP_E_ACCESS_DENIED"
                                                           --+ H-ELEM-ID = 2
                                                           }
                              --+ ATTR-NAME = delete-Error-Info
                              --+ H-ATTR-NAME = "MP_DELETE_ERROR_INFO"
                              --+ H-ATTR-ID = 2029
                              --+ ATTR-SYNTAX = 10 "Delete-Error-Info"
                              --+ VALUE-NUMBER = 1
```

```
                        }


DeleteResult::= --+ SUPER-CLASS = Delete-Result
                --+ CL-NAME = CMIS-Delete-Result
                --+ CL-TYPE = 3
                --+ H-CL-NAME = "OMP_O_MP_C_CMIS_DELETE_RESULT"
                --+ H-CL-ID = 2018
                SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                               --+ ATTR-NAME = managed-Object-Class
                               --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                               --+ H-ATTR-ID = 2057
                               --+ ATTR-SYNTAX = 127 "Object-Class"
                               --+ VALUE-NUMBER = 0
                         ,
                         managedObjectInstance ObjectInstance OPTIONAL
                             --+ ATTR-NAME = managed-Object-Instance
                             --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                             --+ H-ATTR-ID = 2058
                             --+ ATTR-SYNTAX = 127 "Object-Instance"
                             --+ VALUE-NUMBER = 0
                         ,
                         currentTime [5] IMPLICIT GeneralizedTime OPTIONAL
                             --+ ATTR-NAME = current-Time
                             --+ H-ATTR-NAME = "MP_CURRENT_TIME"
                             --+ H-ATTR-ID = 2027
                             --+ ATTR-SYNTAX = 24
                             --+ VALUE-NUMBER = 0
                         }


EventReply::= --+ CL-NAME = Event-Reply
              --+ CL-TYPE = 3
              --+ H-CL-NAME = "OMP_O_MP_C_EVENT_REPLY"
              --+ H-CL-ID = 2034
              SEQUENCE { eventType EventTypeId
                             --+ ATTR-NAME = event-Type
                             --+ H-ATTR-NAME = "MP_EVENT_TYPE"
                             --+ H-ATTR-ID = 2041
                             --+ ATTR-SYNTAX = 127 "Event-Type-Id"
                             --+ VALUE-NUMBER = 1
                       ,
                       eventReplyInfo [8] ANY DEFINED BY eventType
                           --% ANY_TABLE_REF (NotificationReplyTableMod.NotificationReplyTypes)
                                                %-- OPTIONAL
                           --+ ATTR-NAME = event-Reply-Info
                           --+ H-ATTR-NAME = "MP_EVENT_REPLY_INFO"
                           --+ H-ATTR-ID = 2039
                           --+ ATTR-SYNTAX = 8
                           --+ VALUE-NUMBER = 0
                       }


EventReportArgument::= --+ SUPER-CLASS = Event-Report-Argument
                       --+ CL-NAME = CMIS-Event-Report-Argument
                       --+ CL-TYPE = 3
                       --+ H-CL-NAME = "OMP_O_MP_C_CMIS_EVENT_REPORT_ARGUMENT"
                       --+ H-CL-ID = 2019
                       SEQUENCE { managedObjectClass ObjectClass
                                      --+ ATTR-NAME = managed-Object-Class
                                      --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                                      --+ H-ATTR-ID = 2057
                                      --+ ATTR-SYNTAX = 127 "Object-Class"
                                      --+ VALUE-NUMBER = 1
                                ,
                                managedObjectInstance ObjectInstance
                                    --+ ATTR-NAME = managed-Object-Instance
                                    --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                                    --+ H-ATTR-ID = 2058
                                    --+ ATTR-SYNTAX = 127 "Object-Instance"
                                    --+ VALUE-NUMBER = 1
                                ,
                                eventTime [5] IMPLICIT GeneralizedTime OPTIONAL
                                    --+ ATTR-NAME = event-Time
                                    --+ H-ATTR-NAME = "MP_EVENT_TIME"
                                    --+ H-ATTR-ID = 2040
                                    --+ ATTR-SYNTAX = 24
                                    -- Time
                                    --+ VALUE-NUMBER = 0
                                ,
                                eventType EventTypeId
                                    --+ ATTR-NAME = event-Type
                                    --+ H-ATTR-NAME = "MP_EVENT_TYPE"
                                    --+ H-ATTR-ID = 2041
                                    --+ ATTR-SYNTAX = 127 "Event-Type-Id"
                                    --+ VALUE-NUMBER = 1
                                ,
                                eventInfo [8] ANY DEFINED BY eventType
```

```
                                        --% ANY_TABLE_REF (NotificationInfoTableMod.NotificationTypes)
                                                                %-- OPTIONAL
                                        --+ ATTR-NAME = event-Info
                                        --+ H-ATTR-NAME = "MP_EVENT_INFO"
                                        --+ H-ATTR-ID = 2037
                                        --+ ATTR-SYNTAX = 8
                                        --+ VALUE-NUMBER = 0
                                    }


EventReportResult::= --+ SUPER-CLASS = Event-Report-Result
                     --+ CL-NAME = CMIS-Event-Report-Result
                     --+ CL-TYPE = 3
                     --+ H-CL-NAME = "OMP_O_MP_C_CMIS_EVENT_REPORT_RESULT"
                     --+ H-CL-ID = 2020
                     SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                                    --+ ATTR-NAME = managed-Object-Class
                                    --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                                    --+ H-ATTR-ID = 2057
                                    --+ ATTR-SYNTAX = 127 "Object-Class"
                                    --+ VALUE-NUMBER = 0
                                ,
                                managedObjectInstance ObjectInstance OPTIONAL
                                    --+ ATTR-NAME = managed-Object-Instance
                                    --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                                    --+ H-ATTR-ID = 2058
                                    --+ ATTR-SYNTAX = 127 "Object-Instance"
                                    --+ VALUE-NUMBER = 0
                                ,
                                currentTime [5] IMPLICIT GeneralizedTime OPTIONAL
                                    --+ ATTR-NAME = current-Time
                                    --+ H-ATTR-NAME = "MP_CURRENT_TIME"
                                    --+ H-ATTR-ID = 2027
                                    --+ ATTR-SYNTAX = 24
                                    --+ VALUE-NUMBER = 0
                                ,
                                eventReply EventReply OPTIONAL
                                    --+ ATTR-NAME = event-Reply
                                    --+ H-ATTR-NAME = "MP_EVENT_REPLY"
                                    --+ H-ATTR-ID = 2038
                                    --+ ATTR-SYNTAX = 127 "Event-Reply"
                                    --+ VALUE-NUMBER = 0
                                }

-- This has been adapted to align with the comments below.
-- The CHOICE has been eliminated to simplify processing
-- and the use of the API.
EventTypeId ::= [6] IMPLICIT OBJECT IDENTIFIER
-- This [Recommendation | part of ISO/IEC 9596] does not allocate any values for
-- localForm.
-- Where this alternative is used, the permissible values for the integers
-- and their meanings shall be defined as part of the application context
-- in which they are used.

FilterItem::= --+ CL-NAME = Filter-Item
              --+ CL-TYPE = 0
              --+ H-CL-NAME = "OMP_O_MP_C_FILTER_ITEM"
              --+ H-CL-ID = 2036
              CHOICE { equality [0] IMPLICIT Attribute
                           --+ ATTR-NAME = equality
                           --+ H-ATTR-NAME = "MP_EQUALITY"
                           --+ H-ATTR-ID = 2032
                           --+ ATTR-SYNTAX = 127 "Attribute"
                           --+ VALUE-NUMBER = 0
                     ,
                     substrings [1] IMPLICIT SEQUENCE OF Substrings
                         --+ ATTR-NAME = substrings
                         --+ H-ATTR-NAME = "MP_SUBSTRINGS"
                         --+ H-ATTR-ID = 2077
                         --+ ATTR-SYNTAX = 127 "Substrings"
                         --+ VALUE-NUMBER = 2
                     ,
                     greaterOrEqual [2] IMPLICIT Attribute -- asserted value >= attribute value
                         --+ ATTR-NAME = greater-Or-Equal
                         --+ H-ATTR-NAME = "MP_GREATER_OR_EQUAL"
                         --+ H-ATTR-ID = 2050
                         --+ ATTR-SYNTAX = 127 "Attribute"
                         --+ VALUE-NUMBER = 0
                     ,
                     lessOrEqual [3] IMPLICIT Attribute -- asserted value <= attribute value
                         --+ ATTR-NAME = less-Or-Equal
                         --+ H-ATTR-NAME = "MP_LESS_OR_EQUAL"
                         --+ H-ATTR-ID = 2054
                         --+ ATTR-SYNTAX = 127 "Attribute"
                         --+ VALUE-NUMBER = 0
                     ,
                     present [4] AttributeId
                         --+ ATTR-NAME = present
```

```
                        --+ H-ATTR-NAME = "MP_PRESENT"
                        --+ H-ATTR-ID = 2066
                        --+ ATTR-SYNTAX = 127 "Attribute-Id"
                        --+ VALUE-NUMBER = 0
                    ,
                    subsetOf [5] IMPLICIT Attribute -- asserted value is a subset of attribute value
                        --+ ATTR-NAME = subset-Of
                        --+ H-ATTR-NAME = "MP_SUBSET_OF"
                        --+ H-ATTR-ID = 2076
                        --+ ATTR-SYNTAX = 127 "Attribute"
                        --+ VALUE-NUMBER = 0
                    ,
                    supersetOf [6] IMPLICIT Attribute -- asserted value is a superset of attribute value
                        --+ ATTR-NAME = superset-Of
                        --+ H-ATTR-NAME = "MP_SUPERSET_OF"
                        --+ H-ATTR-ID = 2079
                        --+ ATTR-SYNTAX = 127 "Attribute"
                        --+ VALUE-NUMBER = 0
                    ,
                    nonNullSetIntersection [7] IMPLICIT Attribute
                        --+ ATTR-NAME = non-Null-Set-Intersection
                        --+ H-ATTR-NAME = "MP_NON_NULL_SET_INTERSECTION"
                        --+ H-ATTR-ID = 2062
                        --+ ATTR-SYNTAX = 127 "Attribute"
                        --+ VALUE-NUMBER = 0
                    }


    Substring ::= --+ CL-NAME = Substring
                  --+ CL-TYPE = 3
                  --+ H-CL-NAME = "OMP_O_MP_C_SUBSTRING"
                  --+ H-CL-ID = 2053
                  SEQUENCE { attributeId AttributeId
                                --+ ATTR-NAME = attribute-Id
                                --+ H-ATTR-NAME = "MP_ATTRIBUTE_ID"
                                --+ H-ATTR-ID = 2017
                                --+ ATTR-SYNTAX = 127 "Attribute-Id"
                                --+ VALUE-NUMBER = 1
                             ,
                             string ANY DEFINED BY attributeId
                                 --% ANY_TABLE_REF(AttributeTableMod.AttributeTypes)
                                 --+ ATTR-NAME = string
                                 --+ H-ATTR-NAME = "MP_STRING"
                                 --+ H-ATTR-ID = 1039
                                 --+ ATTR-SYNTAX = 27
                                 --+ VALUE-NUMBER = 3
                   }

    Substrings ::= --+ CL-NAME = Substrings
                   --+ CL-TYPE = 4
                   --+ H-CL-NAME = "OMP_O_MP_C_SUBSTRINGS"
                   --+ H-CL-ID = 2054
                   CHOICE { initialString [0] IMPLICIT Substring
                               --+ ATTR-NAME = initial-String
                               --+ H-ATTR-NAME = "MP_INITIAL_STRING"
                               --+ H-ATTR-ID = 2052
                               --+ ATTR-SYNTAX = 127 "Substring"
                               --+ VALUE-NUMBER = 0
                          ,
                          anyString     [1] IMPLICIT Substring
                              --+ ATTR-NAME = any-Substring
                              --+ H-ATTR-NAME = "MP_ANY_STRING"
                              --+ H-ATTR-ID = 2013
                              --+ ATTR-SYNTAX = 127 "Substring"
                              --+ VALUE-NUMBER = 0
                          ,
                          finalString   [2] IMPLICIT Substring
                              --+ ATTR-NAME = final-String
                              --+ H-ATTR-NAME = "MP_FINAL_STRING"
                              --+ H-ATTR-ID = 2044
                              --+ ATTR-SYNTAX = 127 "Substring"
                              --+ VALUE-NUMBER = 0
                        }


GetArgument::= --+ SUPER-CLASS = Get-Argument
               --+ CL-NAME = CMIS-Get-Argument
               --+ CL-TYPE = 3
               --+ H-CL-NAME = "OMP_O_MP_C_CMIS_GET_ARGUMENT"
               --+ H-CL-ID = 2022
               SEQUENCE { baseManagedObjectClass       ObjectClass
                             --+ ATTR-NAME = base-Managed-Object-Class
                             --+ H-ATTR-NAME = "MP_BASE_MANAGED_OBJECT_CLASS"
                             --+ H-ATTR-ID = 2023
                             --+ ATTR-SYNTAX = 127 "Object-Class"
                             --+ VALUE-NUMBER = 1
                         ,
                         baseManagedObjectInstance    ObjectInstance
```

```
                                    --+ ATTR-NAME = base-Managed-Object-Instance
                                    --+ H-ATTR-NAME = "MP_BASE_MANAGED_OBJECT_INSTANCE"
                                    --+ H-ATTR-ID = 2024
                                    --+ ATTR-SYNTAX = 127 "Object-Instance"
                                    --+ VALUE-NUMBER = 1
                                ,
                                accessControl [5] AccessControl OPTIONAL
                                    --+ ATTR-NAME = access-Control
                                    --+ H-ATTR-NAME = "MP_ACCESS_CONTROL"
                                    --+ H-ATTR-ID = 1001
                                    --+ ATTR-SYNTAX = 127 "Access-Control"
                                    --+ VALUE-NUMBER = 0
                                ,
                                synchronization [6] IMPLICIT CMISSync DEFAULT bestEffort
                                    --+ ATTR-NAME = synchronization
                                    --+ H-ATTR-NAME = "MP_SYNCHRONIZATION"
                                    --+ H-ATTR-ID = 2080
                                    --+ ATTR-SYNTAX = 10 "CMIS-Sync"
                                    --+ VALUE-NUMBER = 0
                                ,
                                scope [7] Scope DEFAULT basicScope : baseObject
                                    --+ ATTR-NAME = scope
                                    --+ H-ATTR-NAME = "MP_SCOPE"
                                    --+ H-ATTR-ID = 2070
                                    --+ ATTR-SYNTAX = 127 "Scope"
                                    --+ VALUE-NUMBER = 0
                                ,
                                filter CMISFilter DEFAULT and:{}
                                    --+ ATTR-NAME = filter
                                    --+ H-ATTR-NAME = "MP_FILTER"
                                    --+ H-ATTR-ID = 2043
                                    --+ ATTR-SYNTAX = 127 "CMIS-Filter"
                                    --+ VALUE-NUMBER = 0
                                ,
                                attributeIdList [12] IMPLICIT SET OF AttributeId OPTIONAL
                                    --+ ATTR-NAME = attribute-Id-List
                                    --+ H-ATTR-NAME = "MP_ATTRIBUTE_ID_LIST"
                                    --+ H-ATTR-ID = 2020
                                    --+ ATTR-SYNTAX = 127 "Attribute-Id-List"
                                    --+ VALUE-NUMBER = 0
                                    --+ CL-NAME = Attribute-Id-List
                                    --+ CL-TYPE = 2
                                    --+ H-CL-NAME = "OMP_O_MP_C_ATTRIBUTE_ID_LIST"
                                    --+ H-CL-ID = 2010
                                    --+ ATTR-NAME = attribute-Id
                                    --+ H-ATTR-NAME = "MP_ATTRIBUTE_ID"
                                    --+ H-ATTR-ID = 2017
                                    --+ ATTR-SYNTAX = 127 "Attribute-Id"
                                    --+ VALUE-NUMBER = 2
                                }


GetInfoStatus::= --+ CL-NAME = Get-Info-Status
                 --+ CL-TYPE = 0
                 --+ H-CL-NAME = "OMP_O_MP_C_GET_INFO_STATUS"
                 --+ H-CL-ID = 2037
                 CHOICE { attributeIdError [0] IMPLICIT AttributeIdError
                                    --+ ATTR-NAME = attribute-Id-Error
                                    --+ H-ATTR-NAME = "MP_ATTRIBUTE_ID_ERROR"
                                    --+ H-ATTR-ID = 2019
                                    --+ ATTR-SYNTAX = 127 "Attribute-Id-Error"
                                    --+ VALUE-NUMBER = 0
                                ,
                                attribute [1] IMPLICIT Attribute
                                    --+ ATTR-NAME = attribute
                                    --+ H-ATTR-NAME = "MP_ATTRIBUTE"
                                    --+ H-ATTR-ID = 2015
                                    --+ ATTR-SYNTAX = 127 "Attribute"
                                    --+ VALUE-NUMBER = 0
                                }

GetListError::= --+ SUPER-CLASS = Get-List-Error
                --+ CL-NAME = CMIS-Get-List-Error
                --+ CL-TYPE = 3
                --+ H-CL-NAME = "OMP_O_MP_C_CMIS_GET_LIST_ERROR"
                --+ H-CL-ID = 2023
                SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                                     --+ ATTR-NAME = managed-Object-Class
                                     --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                                     --+ H-ATTR-ID = 2057
                                     --+ ATTR-SYNTAX = 127 "Object-Class"
                                     --+ VALUE-NUMBER = 0
                                 ,
                                 managedObjectInstance ObjectInstance OPTIONAL
                                     --+ ATTR-NAME = managed-Object-Instance
                                     --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                                     --+ H-ATTR-ID = 2058
                                     --+ ATTR-SYNTAX = 127 "Object-Instance"
```

```
                                --+ VALUE-NUMBER = 0
                            ,
                            currentTime [5] IMPLICIT GeneralizedTime OPTIONAL
                                --+ ATTR-NAME = current-Time
                                --+ H-ATTR-NAME = "MP_CURRENT_TIME"
                                --+ H-ATTR-ID = 2027
                                --+ ATTR-SYNTAX = 24
                                --+ VALUE-NUMBER = 0
                            ,
                            getInfoList [6] IMPLICIT SET OF GetInfoStatus
                                --+ ATTR-NAME = get-Info-List
                                --+ H-ATTR-NAME = "MP_GET_INFO_LIST"
                                --+ H-ATTR-ID = 2045
                                --+ ATTR-SYNTAX = 127 "Get-Info-Status"
                                --+ VALUE-NUMBER = 3
                            }


GetResult::= --+ SUPER-CLASS = Get-Result
             --+ CL-NAME = CMIS-Get-Result
             --+ CL-TYPE = 3
             --+ H-CL-NAME = "OMP_O_MP_C_CMIS_GET_RESULT"
             --+ H-CL-ID = 2024
             SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                            --+ ATTR-NAME = managed-Object-Class
                            --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                            --+ H-ATTR-ID = 2057
                            --+ ATTR-SYNTAX = 127 "Object-Class"
                            --+ VALUE-NUMBER = 0
                        ,
                        managedObjectInstance ObjectInstance OPTIONAL
                            --+ ATTR-NAME = managed-Object-Instance
                            --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                            --+ H-ATTR-ID = 2058
                            --+ ATTR-SYNTAX = 127 "Object-Instance"
                            --+ VALUE-NUMBER = 0
                        ,
                        currentTime [5] IMPLICIT GeneralizedTime OPTIONAL
                            --+ ATTR-NAME = current-Time
                            --+ H-ATTR-NAME = "MP_CURRENT_TIME"
                            --+ H-ATTR-ID = 2027
                            --+ ATTR-SYNTAX = 24
                            --+ VALUE-NUMBER = 0
                        ,
                        attributeList [6] IMPLICIT SET OF Attribute OPTIONAL
                            --+ ATTR-NAME = attribute-List
                            --+ H-ATTR-NAME = "MP_ATTRIBUTE_LIST"
                            --+ H-ATTR-ID = 2021
                            --+ ATTR-SYNTAX = 127 "Attribute"
                            --+ VALUE-NUMBER = 2
                        }


InvalidArgumentValue::= --+ CL-NAME = Invalid-Argument-Value
                        --+ CL-TYPE = 0
                        --+ H-CL-NAME = "OMP_O_MP_C_INVALID_ARGUMENT_VALUE"
                        --+ H-CL-ID = 2038
                        CHOICE { actionValue [0] IMPLICIT ActionInfo
                                     --+ ATTR-NAME = action-Value
                                     --+ H-ATTR-NAME = "MP_ACTION_VALUE"
                                     --+ H-ATTR-ID = 2011
                                     --+ ATTR-SYNTAX = 127 "Action-Info"
                                     --+ VALUE-NUMBER = 0
                                 ,
                                 eventValue [1] IMPLICIT SEQUENCE { eventType EventTypeId,
                                                                    eventInfo [8] ANY DEFINED BY eventType
                                                                    --% ANY_TABLE_REF
(NotificationInfoTableMod.NotificationReplyTypes) %-- OPTIONAL }
                                     --+ ATTR-NAME = event-Value
                                     --+ H-ATTR-NAME = "MP_EVENT_VALUE"
                                     --+ H-ATTR-ID = 2042
                                     --+ ATTR-SYNTAX = 127 "Event-Reply"
                                     --+ VALUE-NUMBER = 0
                                 }


LinkedReplyArgument::= --+ SUPER-CLASS = Linked-Reply-Argument
                       --+ CL-NAME = CMIS-Linked-Reply-Argument
                       --+ CL-TYPE = 0
                       --+ H-CL-NAME = "OMP_O_MP_C_CMIS_LINKED_REPLY_ARGUMENT"
                       --+ H-CL-ID = 2025
                       CHOICE { getResult [0] IMPLICIT GetResult
                                    --+ ATTR-NAME = get-Result
                                    --+ H-ATTR-NAME = "MP_GET_RESULT"
                                    --+ H-ATTR-ID = 2048
                                    --+ ATTR-SYNTAX = 127 "CMIS-Get-Result"
                                    --+ VALUE-NUMBER = 0
                                ,
```

```
                                    getListError [1] IMPLICIT GetListError
                                        --+ ATTR-NAME = get-List-Error
                                        --+ H-ATTR-NAME = "MP_GET_LIST_ERROR"
                                        --+ H-ATTR-ID = 2047
                                        --+ ATTR-SYNTAX = 127 "CMIS-Get-List-Error"
                                        --+ VALUE-NUMBER = 0
                                    ,
                                    setResult [2] IMPLICIT SetResult
                                        --+ ATTR-NAME = set-Result
                                        --+ H-ATTR-NAME = "MP_SET_RESULT"
                                        --+ H-ATTR-ID = 2074
                                        --+ ATTR-SYNTAX = 127 "CMIS-Set-Result"
                                        --+ VALUE-NUMBER = 0
                                    ,
                                    setListError [3] IMPLICIT SetListError
                                        --+ ATTR-NAME = set-List-Error
                                        --+ H-ATTR-NAME = "MP_SET_LIST_ERROR"
                                        --+ H-ATTR-ID = 2072
                                        --+ ATTR-SYNTAX = 127 "CMIS-Set-List-Error"
                                        --+ VALUE-NUMBER = 0
                                    ,
                                    actionResult [4] IMPLICIT ActionResult
                                        --+ ATTR-NAME = action-Result
                                        --+ H-ATTR-NAME = "MP_ACTION_RESULT"
                                        --+ H-ATTR-ID = 2009
                                        --+ ATTR-SYNTAX = 127 "CMIS-Action-Result"
                                        --+ VALUE-NUMBER = 0
                                    ,
                                    processingFailure [5] IMPLICIT ProcessingFailure
                                        --+ ATTR-NAME = processing-Failure
                                        --+ H-ATTR-NAME = "MP_PROCESSING_FAILURE"
                                        --+ H-ATTR-ID = 2067
                                        --+ ATTR-SYNTAX = 127 "Processing-Failure"
                                        --+ VALUE-NUMBER = 0
                                    ,
                                    deleteResult [6] IMPLICIT DeleteResult
                                        --+ ATTR-NAME = delete-Result
                                        --+ H-ATTR-NAME = "MP_DELETE_RESULT"
                                        --+ H-ATTR-ID = 2030
                                        --+ ATTR-SYNTAX = 127 "CMIS-Delete-Result"
                                        --+ VALUE-NUMBER = 0
                                    ,
                                    actionError [7] IMPLICIT ActionError
                                        --+ ATTR-NAME = action-Error
                                        --+ H-ATTR-NAME = "MP_ACTION_ERROR"
                                        --+ H-ATTR-ID = 2002
                                        --+ ATTR-SYNTAX = 127 "Action-Error"
                                        --+ VALUE-NUMBER = 0
                                    ,
                                    deleteError [8] IMPLICIT DeleteError
                                        --+ ATTR-NAME = delete-Error
                                        --+ H-ATTR-NAME = "MP_DELETE_ERROR"
                                        --+ H-ATTR-ID = 2028
                                        --+ ATTR-SYNTAX = 127 "Delete-Error"
                                        --+ VALUE-NUMBER = 0
                                    }


ModifyOperator::= INTEGER --+ VL-NAME = Modify-Operator
                          { replace (0)
                            --+ ELEM-NAME = replace
                            --+ H-ELEM-NAME = "MP_T_REPLACE"
                            --+ H-ELEM-ID = 0
                            ,
                            addValues (1)
                            --+ ELEM-NAME = add-Values
                            --+ H-ELEM-NAME = "MP_T_ADD_VALUES"
                            --+ H-ELEM-ID = 1
                            ,
                            removeValues (2)
                            --+ ELEM-NAME = remove-Values
                            --+ H-ELEM-NAME = "MP_T_REMOVE_VALUES"
                            --+ H-ELEM-ID = 2
                            ,
                            setToDefault (3)
                            --+ ELEM-NAME = set-To-Default
                            --+ H-ELEM-NAME = "MP_T_SET_TO_DEFAULT"
                            --+ H-ELEM-ID = 3
                            }

NoSuchAction::= --+ CL-NAME = No-Such-Action
                --+ CL-TYPE = 3
                --+ H-CL-NAME = "OMP_O_MP_C_NO_SUCH_ACTION"
                --+ H-CL-ID = 2042
                SEQUENCE { managedObjectClass ObjectClass
                               --+ ATTR-NAME = managed-Object-Class
                               --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                               --+ H-ATTR-ID = 2057
```

```
                                --+ ATTR-SYNTAX = 127 "Object-Class"
                                --+ VALUE-NUMBER = 1
                            ,
                            actionType ActionTypeId
                                --+ ATTR-NAME = action-Type
                                --+ H-ATTR-NAME = "MP_ACTION_TYPE"
                                --+ H-ATTR-ID = 2010
                                --+ ATTR-SYNTAX = 127 "Action-Type-Id"
                                --+ VALUE-NUMBER = 1
                            }


NoSuchArgument::= --+ CL-NAME = No-Such-Argument
                  --+ CL-TYPE = 0
                  --+ H-CL-NAME = "OMP_O_MP_C_NO_SUCH_ARGUMENT"
                  --+ H-CL-ID = 2044
                  CHOICE { actionId [0] IMPLICIT SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                                                                --+ ATTR-NAME = managed-Object-Class
                                                                --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                                                                --+ H-ATTR-ID = 2057
                                                                --+ ATTR-SYNTAX = 127 "Object-Class"
                                                                --+ VALUE-NUMBER = 0
                                                           ,
                                                           actionType ActionTypeId
                                                                --+ ATTR-NAME = action-Type
                                                                --+ H-ATTR-NAME = "MP_ACTION_TYPE"
                                                                --+ H-ATTR-ID = 2010
                                                                --+ ATTR-SYNTAX = 127 "Action-Type-Id"
                                                                --+ VALUE-NUMBER = 1
                                                           }
                                --+ ATTR-NAME = action-Id
                                --+ H-ATTR-NAME = "MP_ACTION_ID"
                                --+ H-ATTR-ID = 2004
                                --+ ATTR-SYNTAX = 127 "No-Such-Action-Id"
                                --+ VALUE-NUMBER = 0
                                --+ CL-NAME = No-Such-Action-Id
                                --+ CL-TYPE = 3
                                --+ H-CL-NAME = "OMP_O_MP_C_NO_SUCH_ACTION_ID"
                                --+ H-CL-ID = 2043
                            ,
                            eventId [1] IMPLICIT SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                                                                --+ ATTR-NAME = managed-Object-Class
                                                                --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                                                                --+ H-ATTR-ID = 2057
                                                                --+ ATTR-SYNTAX = 127 "Object-Class"
                                                                --+ VALUE-NUMBER = 0
                                                           ,
                                                           eventType EventTypeId
                                                                --+ ATTR-NAME = event-Type
                                                                --+ H-ATTR-NAME = "MP_EVENT_TYPE"
                                                                --+ H-ATTR-ID = 2041
                                                                --+ ATTR-SYNTAX = 127 "Event-Type-Id"
                                                                --+ VALUE-NUMBER = 1
                                                           }
                                --+ ATTR-NAME = event-Id
                                --+ H-ATTR-NAME = "MP_EVENT_ID"
                                --+ H-ATTR-ID = 2036
                                --+ ATTR-SYNTAX = 127 "No-Such-Event-Id"
                                --+ VALUE-NUMBER = 0
                                --+ CL-NAME = No-Such-Event-Id
                                --+ CL-TYPE = 3
                                --+ H-CL-NAME = "OMP_O_MP_C_NO_SUCH_EVENT_ID"
                                --+ H-CL-ID = 2045
                            }

NoSuchEventType::= --+ CL-NAME = No-Such-Event-Type
                   --+ CL-TYPE = 3
                   --+ H-CL-NAME = "OMP_O_MP_C_NO_SUCH_EVENT_TYPE"
                   --+ H-CL-ID = 2046
                   SEQUENCE { managedObjectClass ObjectClass
                                   --+ ATTR-NAME = managed-Object-Class
                                   --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                                   --+ H-ATTR-ID = 2057
                                   --+ ATTR-SYNTAX = 127 "Object-Class"
                                   --+ VALUE-NUMBER = 1
                              ,
                              eventType EventTypeId
                                   --+ ATTR-NAME = event-Type
                                   --+ H-ATTR-NAME = "MP_EVENT_TYPE"
                                   --+ H-ATTR-ID = 2041
                                   --+ ATTR-SYNTAX = 127 "Event-Type-Id"
                                   --+ VALUE-NUMBER = 1
                              }


-- ADDED to allow support for allomorphic Notifications
-- The first production is used by the Infrastructure to parse the
--   notification as a complete unit. This gets around the resolution
```

```
--   of the argument to EventReportArgument in the ROIV production.
-- The second production is used in the place of EventReportArgument
-- when applications wish to emit notifications
Notification ::= [1] IMPLICIT SEQUENCE
   {invokeID        InvokeIDType,
    linked-ID   [0] IMPLICIT InvokeIDType OPTIONAL,
    operation-value Operation,
    argument        NotificationArg
    }

NotificationArg ::= SEQUENCE
   {
   allomorphs    SET OF ObjectClass OPTIONAL,
    managedObjectClass      ObjectClass,
    managedObjectInstance   ObjectInstance,
    eventTime           [5] IMPLICIT GeneralizedTime OPTIONAL,
    eventType               EventTypeId,
    eventInfo           [8] ANY DEFINED BY eventType
    --% ANY_TABLE_REF (NotificationInfoTableMod.NotificationTypes) %-- OPTIONAL
   }
-- End of Notification addition


-- This is actually a CHOICE of OBJECT IDENTIFIER or INTEGER
-- but we only support OBJECT IDENTIFIER, so the syntax was simplified
-- to shorten the strings at the API
ObjectClass ::= [0] IMPLICIT OBJECT IDENTIFIER
-- This [Recommendation | part of ISO/IEC 9596] does not allocate any values for
-- localForm.
-- Where this alternative is used, the permissible values for the integers
-- and their meanings shall be defined as part of the application context
-- in which they are used.

ObjectInstance::= --+ CL-NAME = Object-Instance
                  --+ CL-TYPE = 0
                  --+ H-CL-NAME = "OMP_O_MP_C_OBJECT_INSTANCE"
                  --+ H-CL-ID = 2048
                  CHOICE { distinguishedName [2] IMPLICIT DistinguishedName
                               --+ ATTR-NAME = distinguished-Name
                               --+ H-ATTR-NAME = "MP_DISTINGUISHED_NAME"
                               --+ H-ATTR-ID = 2031
                               --+ ATTR-SYNTAX = 127 "DS-DN"
                               --+ VALUE-NUMBER = 0
                         ,
                         nonSpecificForm [3] IMPLICIT OCTET STRING
                             --+ ATTR-NAME = non-Specific-Form
                             --+ H-ATTR-NAME = "MP_NON_SPECIFIC_FORM"
                             --+ H-ATTR-ID = 2063
                             --+ ATTR-SYNTAX = 4
                             --+ VALUE-NUMBER = 0
                         ,
                         localDistinguishedName [4] IMPLICIT RDNSequence
                             --+ ATTR-NAME = local-DN
                             --+ H-ATTR-NAME = "MP_LOCAL_DN"
                             --+ H-ATTR-ID = 2055
                             --+ ATTR-SYNTAX = 127 "DS-DN"
                             --+ VALUE-NUMBER = 0
                         }

-- localDistinguishedName is that portion of the distinguished name that is
-- necessary to unambiguosly identify the managed object within the context
-- of communication between the open systems

ProcessingFailure::= --+ CL-NAME = Processing-Failure
                     --+ CL-TYPE = 3
                     --+ H-CL-NAME = "OMP_O_MP_C_PROCESSING_FAILURE"
                     --+ H-CL-ID = 2049
                     SEQUENCE { managedObjectClass ObjectClass
                                  --+ ATTR-NAME = managed-Object-Class
                                  --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                                  --+ H-ATTR-ID = 2057
                                  --+ ATTR-SYNTAX = 127 "Object-Class"
                                  --+ VALUE-NUMBER = 1
                              ,
                              managedObjectInstance ObjectInstance OPTIONAL
                                  --+ ATTR-NAME = managed-Object-Instance
                                  --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                                  --+ H-ATTR-ID = 2058
                                  --+ ATTR-SYNTAX = 127 "Object-Instance"
                                  --+ VALUE-NUMBER = 0
                              ,
                              specificErrorInfo [5] SpecificErrorInfo
                                  --+ ATTR-NAME = specific-Error-Info
                                  --+ H-ATTR-NAME = "MP_SPECIFIC_ERROR_INFO"
                                  --+ H-ATTR-ID = 2075
                                  --+ ATTR-SYNTAX = 127 "Specific-Error-Info"
                                  --+ VALUE-NUMBER = 1
                              }
```

```
Scope::= --+ CL-NAME = Scope
         --+ CL-TYPE = 0
         --+ H-CL-NAME = "OMP_O_MP_C_SCOPE"
         --+ H-CL-ID = 2050
         CHOICE { basicScope  INTEGER --+ VL-NAME = Scope
                         { baseObject (0)
                           --+ ELEM-NAME = base-Object
                           --+ H-ELEM-NAME = "MP_T_BASE_OBJECT"
                           --+ H-ELEM-ID = 0
                           ,
                           firstLevelOnly (1)
                           --+ ELEM-NAME = first-Level-Only
                           --+ H-ELEM-NAME = "MP_T_FIRST_LEVEL_ONLY"
                           --+ H-ELEM-ID = 1
                           ,
                           wholeSubtree (2)
                           --+ ELEM-NAME = whole-Subtree
                           --+ H-ELEM-NAME = "MP_T_WHOLE_SUBTREE"
                           --+ H-ELEM-ID = 2
                           }
                      --+ ATTR-NAME = named-Numbers
                      --+ H-ATTR-NAME = "MP_NAMED_NUMBERS"
                      --+ H-ATTR-ID = 2061
                      --+ ATTR-SYNTAX = 10 "Scope"
                      --+ VALUE-NUMBER = 0
                  ,
                  individualLevels [1] IMPLICIT INTEGER
                      -- POSITIVE integer that indicates the level to be selected
                      --+ ATTR-NAME = individual-Levels
                      --+ H-ATTR-NAME = "MP_INDIVIDUAL_LEVELS"
                      --+ H-ATTR-ID = 2051
                      --+ ATTR-SYNTAX = 2
                      --+ VALUE-NUMBER = 0
                  ,
                  baseToNthLevel   [2] IMPLICIT INTEGER
                      -- POSITIVE integer that indicates the range of levels (0-N) is to be selected
                      --+ ATTR-NAME = base-To-Nth-Level
                      --+ H-ATTR-NAME = "MP_BASE_TO_NTH_LEVEL"
                      --+ H-ATTR-ID = 2025
                      --+ ATTR-SYNTAX = 2
                      --+ VALUE-NUMBER = 0
                  }

-- with individualLevels and baseToNthLevel, a value of 0 has the same semantics
-- as baseObject
-- with individualLevels, a value of 1 has the same semantics as firstLevelOnly


SetArgument::= --+ SUPER-CLASS = Set-Argument
               --+ CL-NAME = CMIS-Set-Argument
               --+ CL-TYPE = 3
               --+ H-CL-NAME = "OMP_O_MP_C_CMIS_SET_ARGUMENT"
               --+ H-CL-ID = 2027
               SEQUENCE { baseManagedObjectClass       ObjectClass
                               --+ ATTR-NAME = base-Managed-Object-Class
                               --+ H-ATTR-NAME = "MP_BASE_MANAGED_OBJECT_CLASS"
                               --+ H-ATTR-ID = 2023
                               --+ ATTR-SYNTAX = 127 "Object-Class"
                               --+ VALUE-NUMBER = 1
                          ,
                          baseManagedObjectInstance  ObjectInstance
                              --+ ATTR-NAME = base-Managed-Object-Instance
                              --+ H-ATTR-NAME = "MP_BASE_MANAGED_OBJECT_INSTANCE"
                              --+ H-ATTR-ID = 2024
                              --+ ATTR-SYNTAX = 127 "Object-Instance"
                              --+ VALUE-NUMBER = 1
                          ,
                          accessControl [5] AccessControl OPTIONAL
                              --+ ATTR-NAME = access-Control
                              --+ H-ATTR-NAME = "MP_ACCESS_CONTROL"
                              --+ H-ATTR-ID = 1001
                              --+ ATTR-SYNTAX = 127 "Access-Control"
                              --+ VALUE-NUMBER = 0
                          ,
                          synchronization [6] IMPLICIT CMISSync DEFAULT bestEffort
                              --+ ATTR-NAME = synchronization
                              --+ H-ATTR-NAME = "MP_SYNCHRONIZATION"
                              --+ H-ATTR-ID = 2080
                              --+ ATTR-SYNTAX = 10 "CMIS-Sync"
                              --+ VALUE-NUMBER = 0
                          ,
                          scope [7] Scope DEFAULT basicScope : baseObject
                              --+ ATTR-NAME = scope
                              --+ H-ATTR-NAME = "MP_SCOPE"
                              --+ H-ATTR-ID = 2070
                              --+ ATTR-SYNTAX = 127 "Scope"
```

```
                                --+ VALUE-NUMBER = 0
                              ,
                              filter CMISFilter DEFAULT and:{}
                                --+ ATTR-NAME = filter
                                --+ H-ATTR-NAME = "MP_FILTER"
                                --+ H-ATTR-ID = 2043
                                --+ ATTR-SYNTAX = 127 "CMIS-Filter"
                                --+ VALUE-NUMBER = 0
                              ,
                              modificationList [12] IMPLICIT SET OF
                                --+ CL-NAME = Modification
                                --+ CL-TYPE = 3
                                --+ H-CL-NAME = "OMP_O_MP_C_MODIFICATION"
                                --+ H-CL-ID = 2040
                                SEQUENCE { modifyOperator [2]
IMPLICIT ModifyOperator DEFAULT replace
                                --+ ATTR-NAME = modify-Operator
                                --+ H-ATTR-NAME = "MP_MODIFY_OPERATOR"
                                --+ H-ATTR-ID = 2060
                                --+ ATTR-SYNTAX = 10 "Modify-Operator"
                                --+ VALUE-NUMBER = 0
                             ,
                             attributeId AttributeId
                                --+ ATTR-NAME = attribute-Id
                                --+ H-ATTR-NAME = "MP_ATTRIBUTE_ID"
                                --+ H-ATTR-ID = 2017
                                --+ ATTR-SYNTAX = 127 "Attribute-Id"
                                --+ VALUE-NUMBER = 1
                             ,
                             attributeValue ANY DEFINED BY attributeId
                                --% ANY_TABLE_REF(AttributeTableMod.AttributeTypes)
%-- OPTIONAL -- absent for setToDefault
                                --+ ATTR-NAME = attribute-Value
                                --+ H-ATTR-NAME = "MP_ATTRIBUTE_VALUE"
                                --+ H-ATTR-ID = 2022
                                --+ ATTR-SYNTAX = 8
                                --+ VALUE-NUMBER = 0
                              }
                                --+ ATTR-NAME = modification-List
                                --+ H-ATTR-NAME = "MP_MODIFICATION_LIST"
                                --+ H-ATTR-ID = 2059
                                --+ ATTR-SYNTAX = 127 "Modification"
                                --+ VALUE-NUMBER = 2
                              }


SetInfoStatus::= --+ CL-NAME = Set-Info-Status
                 --+ CL-TYPE = 0
                 --+ H-CL-NAME = "OMP_O_MP_C_SET_INFO_STATUS"
                 --+ H-CL-ID = 2051
                 CHOICE { attributeError [0] IMPLICIT AttributeError
                                --+ ATTR-NAME = attribute-Error
                                --+ H-ATTR-NAME = "MP_ATTRIBUTE_ERROR"
                                --+ H-ATTR-ID = 2016
                                --+ ATTR-SYNTAX = 127 "Attribute-Error"
                                --+ VALUE-NUMBER = 0
                              ,
                              attribute [1] IMPLICIT Attribute
                                --+ ATTR-NAME = attribute
                                --+ H-ATTR-NAME = "MP_ATTRIBUTE"
                                --+ H-ATTR-ID = 2015
                                --+ ATTR-SYNTAX = 127 "Attribute"
                                --+ VALUE-NUMBER = 0
                              }


SetListError::= --+ SUPER-CLASS = Set-List-Error
                --+ CL-NAME = CMIS-Set-List-Error
                --+ CL-TYPE = 3
                --+ H-CL-NAME = "OMP_O_MP_C_CMIS_SET_LIST_ERROR"
                --+ H-CL-ID = 2028
                SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                                --+ ATTR-NAME = managed-Object-Class
                                --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                                --+ H-ATTR-ID = 2057
                                --+ ATTR-SYNTAX = 127 "Object-Class"
                                --+ VALUE-NUMBER = 0
                             ,
                             managedObjectInstance ObjectInstance OPTIONAL
                                --+ ATTR-NAME = managed-Object-Instance
                                --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                                --+ H-ATTR-ID = 2058
                                --+ ATTR-SYNTAX = 127 "Object-Instance"
                                --+ VALUE-NUMBER = 0
                             ,
                             currentTime [5] IMPLICIT GeneralizedTime OPTIONAL
                                --+ ATTR-NAME = current-Time
                                --+ H-ATTR-NAME = "MP_CURRENT_TIME"
```

```
                                --+ H-ATTR-ID = 2027
                                --+ ATTR-SYNTAX = 24
                                --+ VALUE-NUMBER = 0
                            ,
                            setInfoList [6] IMPLICIT SET OF SetInfoStatus
                                --+ ATTR-NAME = set-Info-List
                                --+ H-ATTR-NAME = "MP_SET_INFO_LIST"
                                --+ H-ATTR-ID = 2071
                                --+ ATTR-SYNTAX = 127 "Set-Info-Status"
                                --+ VALUE-NUMBER = 3
                            }


SetResult::= --+ SUPER-CLASS = Set-Result
             --+ CL-NAME = CMIS-Set-Result
             --+ CL-TYPE = 3
             --+ H-CL-NAME = "OMP_O_MP_C_CMIS_SET_RESULT"
             --+ H-CL-ID = 2029

             SEQUENCE { managedObjectClass ObjectClass OPTIONAL
                            --+ ATTR-NAME = managed-Object-Class
                            --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_CLASS"
                            --+ H-ATTR-ID = 2057
                            --+ ATTR-SYNTAX = 127 "Object-Class"
                            --+ VALUE-NUMBER = 0
                        ,
                        managedObjectInstance ObjectInstance OPTIONAL
                            --+ ATTR-NAME = managed-Object-Instance
                            --+ H-ATTR-NAME = "MP_MANAGED_OBJECT_INSTANCE"
                            --+ H-ATTR-ID = 2058
                            --+ ATTR-SYNTAX = 127 "Object-Instance"
                            --+ VALUE-NUMBER = 0
                        ,
                        currentTime [5] IMPLICIT GeneralizedTime OPTIONAL
                            --+ ATTR-NAME = current-Time
                            --+ H-ATTR-NAME = "MP_CURRENT_TIME"
                            --+ H-ATTR-ID = 2027
                            --+ ATTR-SYNTAX = 24
                            --+ VALUE-NUMBER = 0
                        ,
                        attributeList [6] IMPLICIT SET OF Attribute OPTIONAL
                            --+ ATTR-NAME = attribute-List
                            --+ H-ATTR-NAME = "MP_ATTRIBUTE_LIST"
                            --+ H-ATTR-ID = 2021
                            --+ ATTR-SYNTAX = 127 "Attribute"
                            --+ VALUE-NUMBER = 2
                        }

SpecificErrorInfo::= --+ CL-NAME = Specific-Error-Info
                     --+ CL-TYPE = 3
                     --+ H-CL-NAME = "OMP_O_MP_C_SPECIFC_ERROR_INFO"
                     --+ H-CL-ID = 2052
                     SEQUENCE { errorId OBJECT IDENTIFIER
                                    --+ ATTR-NAME = error-Id
                                    --+ H-ATTR-NAME = "MP_ERROR_ID"
                                    --+ H-ATTR-ID = 2033
                                    --+ ATTR-SYNTAX = 6
                                    --+ VALUE-NUMBER = 1
                                ,
                                errorInfo ANY DEFINED BY errorId --% ANY_TABLE_REF(ParameterTableMod.
                                                                                   ParameterTypes)
                                    --+ ATTR-NAME = error-Info
                                    --+ H-ATTR-NAME = "MP_ERROR_INFO"
                                    --+ H-ATTR-ID = 2034
                                    --+ ATTR-SYNTAX = 8
                                    --+ VALUE-NUMBER = 1
                                }


END -- CMIP definitions
```

Appendix C. Error codes sent by CMIP services

This appendix includes descriptions of error codes from ACYAPHDH, which is shipped in the AMACLIB data set of the SYS1.MACLIB data set. These errors can be received in the following:

- MIB.ServiceError strings
- CMER VTAM internal trace (VIT) entries

MIB.ServiceError error codes

These errors can be received in MIB.ServiceError strings.

0 (Indicates success.)

Explanation: This is used to denote normal, correct processing.

Action: None, everything is working correctly.

7 MB_ERR_ALLOC

Explanation: An attempt was made to allocate memory for the processing of a message. The operating system returned an error. The platform will halt processing of this message and attempt to recover. This message will be lost. If the condition was transient, all later messages may work correctly.

Action: This should occur only if the system is reaching a private or common storage area (CSA) storage limit. If this is encountered and does not seem to be transient, you will need to increase the limit causing the problem. If this error is received by the application program from an API function and there is a corresponding REQS record in the VIT with a nonzero return code, the LPBUF pool is not large enough and should be increased.

8 PROGRAM_CHECK

Explanation: A condition that should not be able to happen has occurred.

Action: Call IBM Service. Please provide the error log and as much information about what was being processed as possible. This includes:

- The trace, if one exists
- The message being processed
- The set of instantiated objects
- The list of associations
- The set of outstanding CMIP operations

250 AUTHENTICATION_FAILED

Explanation: The association could not be established due to security.

Action: Consult the directory definition file on both systems to resolve inconsistencies.

251 AUTHENTICATION_INFO_MISSING

Explanation: Either data encryption standard (DES) based security or application-program-to-application-program security is required. The association could not be established due to security.

252 AUTHENTICATION_MECH_UNKNOWN

Explanation: There is a mismatch in the ASN.1 for the association request between the two systems.

Action: Call IBM Service.

300 BER_BAD_TYPE

Explanation: The encode/decode functions of CMIP services were called to parse a message. They were told to parse it using the syntax defined as an identified module and type. The module is one that is loaded, the type name is not.

Action: If the message being processed is a CMIP message, and the application program uses only MIBSendCmipRequest and MIBSendCmipResponse, call IBM Service. If the message being parsed was one that was passed to the platform with MIBSendServiceRequest or MIBSendRequest, correct the type name in the message to be one that is contained in the indicated module.

301 BER_BAD_MODULE

Explanation: The encode/decode functions of CMIP services were called to parse a message. They were told to parse it using the syntax defined as an identified module and type. The module is one that is not loaded.

Action: If the message being processed is a CMIP message, and the application program uses only MIBSendCmipRequest and MIBSendCmipResponse, call IBM Service. If the message being parsed was one that was passed to the platform with MIBSendServiceRequest or MIBSendRequest, correct the module name in the message.

302 BER_NULL_TYPE

Explanation: The portion of CMIP services that calls the encode/decode functions passed in a NULL type name to be processed.

Action: Call IBM Service.

303 BER_NULL_MODULE

Explanation: The portion of CMIP services that calls the encode/decode functions passed in a NULL module name to be processed.

Action: Call IBM Service.

304 BER_NULL_STRING

Explanation: The string passed to the encode/decode component of CMIP services was NULL.

Action: Call IBM Service

305 BER_NULL_STRUCT

Explanation: The data structure passed to the encode/decode component of CMIP services to contain the result was NULL.

Action: Call IBM Service.

306 BER_BAD_METATABLE

Explanation: The ASN.1 data set is not correct.

Action: Reload the ISTASN1 data set from the distribution media.

307 BER_UNKNOWN_TYPE

Explanation: The data type derived for a node in the tree constructed while parsing a message is unrecognized. Since these are the base types defined in ASN.1. this should not happen.

Action: Call IBM Service.

308 BER_UNKNOWN_MEMBER

Explanation: While processing a SET or SEQUENCE, the encode/decode component of CMIP services encountered an element that did not belong in the SET or SEQUENCE.

Action: If this occurred while processing a string value generated by the application, correct the application program to send a valid value. If this occurred while processing a received BER buffer from a peer entity, the syntaxes on the two systems do not match. You will need to analyze the differences, determine which version is correct and load the corrected syntaxes on one or both systems.

309 BER_UNKNOWN_ALTERNATIVE

Explanation: While processing a CHOICE, the encode/decode component of CMIP services encountered an element that did not represent one of the valid alternatives for the CHOICE.

Action: If this occurred while processing a string value generated by the application, correct the application program to send a valid value. If this occurred while processing a received BER buffer from a peer entity, the syntaxes on the two systems do not match. You will need to analyze the differences, determine which version is correct and load the corrected syntaxes on one or both systems.

310 BER_NO_END_PARENTHESIS

Explanation: While parsing the string message from an application program CMIP services determined that there should have been a closing parenthesis at the indicated location in the string. This represents the end of a SET, SEQUENCE or CHOICE.

Action: Correct the string message.

311 BER_NO_START_PARENTHESIS

Explanation: While parsing the string message from an application program CMIP services determined that there should have been an opening parenthesis at the indicated location in the string. This represents the beginning of a SET, SEQUENCE or CHOICE.

Action: Correct the string message.

312 BER_NO_MORE_STRING

Explanation: Additional information was expected in the string buffer. The buffer terminated prematurely. There were either missing mandatory elements or (at least) some missing closing parentheses in the string.

Action: Correct the string value.

313 BER_PARSE_ERROR

Explanation: An error occurred during the parsing of the message. The message is invalid. This error is only issued when no more specific error is encountered.

Action: Correct the string value.

314 BER_IMPLICIT_CHOICE

Explanation: While parsing the string message the encode/decode component of CMIP services encountered an IMPLICIT CHOICE. This is not legal in ASN.1. This should have been caught and converted to an EXPLICIT CHOICE by the ASN.1 compiler.

Action: Correct the string value.

315 BER_CANNOT_RESOLVE

Explanation: An ANY DEFINED BY was encountered for which an ANY TABLE was defined. This table defines all of the values that will be understood for ANY DEFINED BY resolution. The table did not include the value provided.

Action: Correct the value in the string or add the value to the ANY DEFINED BY table.

316 BER_NEED_LABEL

Explanation: Either a CHOICE was encountered where the user tried to omit the label of the chosen alternative, or a SET was encountered in which the user failed to specify the label of an element. Both of these require labels in order to provide unambiguous resolution of the string.

Action: Add the required label to the string.

317 BER_MISSING_MEMBER

Explanation: A mandatory element was omitted from a SET or SEQUENCE, or the label for the mandatory element was misspelled.

Action: Correct the string.

319 BER_NO_PARENT

Explanation: In order to resolve an ANY DEFINED BY it is necessary to find the element of the syntax that contains the value to be used to do the resolution. Since ASN.1 requires that this be a mandatory member of the same SEQUENCE, the parsing code goes to the "parent" of the ANY DEFINED BY and searches for the resolution node. The ANY DEFINED BY did not contain a valid reference to a parent.

Action: Call IBM Service.

320 BER_BAD_DN_PARSE

Explanation: While parsing a DistinguishedName some kind of error occurred. This error will be logged before the log of BER_BAD_DN_PARSE. This error serves to narrow the investigation to a DN if the problem is difficult to isolate.

Action: See the previously logged errors and fix the error in the syntax of the name.

321 BER_BAD_RESOLUTION_NODE

Explanation: There are only two data types that can be used to provide resolution for an ANY DEFINED BY. These are INTEGER and OBJECT IDENTIFIER. A case was found while processing this message where the element of the syntax being used for ANY DEFINED BY resolution was another data type.

Action: Correct the syntax of the resolution node in the ASN.1 syntax. If the syntax is determined to be correct, call IBM Service.

322 BER_MISSING_RESOLUTION_NODE

Explanation: An ANY DEFINED BY was encountered while processing the message that does not reference an ANY TABLE to allow resolution of data types. If the message is being decoded, any application program to which the message is sent will receive the contents of this ANY DEFINED BY in BER. This is probably not what the application program s are expecting.

Action: Add an ANY TABLE REF and ANY TABLE to this syntax in the ASN.1.

323 BER_LABEL_MISMATCH

Explanation: The string being parsed by the encode/decode component of CMIP services contains an initial label that does not match any of the possible initial labels for the module.type being parsed.

Action: Correct the string.

325 BER_NOT_BOOLEAN

Explanation: The accepted BOOLEAN values are 'TRUE', 'true', 'FALSE', 'false' and any ASN.1 value references that are defined to be BOOLEAN values.

Action: Correct the value in the string.

326 BER_NOT_INTEGER

Explanation: The accepted INTEGER values are composed of digits optionally prepended with a '+' or '-' sign character, or a defined ASN.1 value reference.

Action: Correct the value in the string.

327 BER_NOT_REAL

Explanation: The syntax expected for a REAL value is exactly the syntax accepted by the C standard function scanf using the format string "%lg". Anything else will be rejected.

Action: Correct the value in the string.

328 BER_NOT_NULL

Explanation: The accepted NULL values are 'NULL', 'null' or a defined ASN.1 value reference to a value of type NULL.

Action: Correct the value in the string.

329 BER_NOT_BIT_STRING

Explanation: The accepted BIT STRING values are composed of zero or more '1' and '0' characters or a value reference to a value of the type BIT STRING.

Action: Correct the value in the string.

330 BER_NOT_HEX_STRING

Explanation: The value being parsed as an OCTET STRING was composed of characters other than legal hexadecimal digits so it was assumed to be a value reference. The value reference was not found.

Action: Correct the value in the string.

331 BER_BAD_HEX_STRING

Explanation: The value specified for an OCTET STRING was not a legal value. It must be an even number of hex digits.

Action: Correct the value in the string.

332 BER_NOT_OI

Explanation: The value encountered for an OBJECT IDENTIFIER does not conform to the dotted-decimal notation and is not a value reference. All values for OBJECT IDENTIFIERs must be composed of digits and periods, and they must contain at least 2 components. Legal values: '1.3.18.0.2.4.5', '1.2' Illegal values: '1', 'joint-iso-ccitt.9.3.2.7.4'

Action: Correct the value in the string.

333 BER_BAD_TIME

Explanation: If a string is being processed, the value specified did not conform to the format specified for times in the string API documentation. If a BER-encoded buffer is being processed, the value does not represent a valid time in BER format.

Action: Correct the value in the string.

334 BER_BAD_ENUMERATED

Explanation: The value encountered was not a valid ENUMERATED.

Action: If the value being processed is a string, correct the string. If the value is a BER buffer, the syntaxes understood by the two systems are different. Align the syntax definitions on the two systems.

335 BER_BAD_PRINTABLE_STRING

Explanation: The value encountered was defined to be a PrintableString. It contained characters that are not allowed in the specification of the PrintableString type. The allowed values for Printable string are:

```
A-Z, a-z, 0-9, space, '\', '(', '0', '+',
',', '-', '.', '/', ':', '=' and '?'
```

Action: Correct the value to be a valid PrintableString.

336 BER_BAD_NUMERIC_STRING

Explanation: The value encountered was not a valid NumericString. NumericStrings can only contain digits and spaces.

Action: Correct the value to be a valid NumericString.

337 BER_BAD_VISIBLE_STRING

Explanation: The value encountered was defined to be of type VisibleString. It contained one or more characters that are not allowed in this data type. The allowed characters are: A-Z, a-z, space and punctuation.

Action: Correct the value to be a valid VisibleString.

338 BER_BAD_GRAPHIC_STRING

Explanation: The value encountered for a GraphicString contained a character that is not presently supported by CMIP services for GraphicString. At the present time the platform only supports the printable characters in a normal ASCII character set.

Action: Correct the value to be within the set supported by the platform.

339 BER_BAD_GENERAL_STRING

Explanation: The value encountered was not a valid general string.

Action: Correct the value.

340 BER_BAD_IA5_STRING

Explanation: The value encountered was not a valid IA5 string.

Action: Correct the value.

341 BER_DUPLICATE_MEMBER

Explanation: A SET is allowed to contain each element only once. While parsing the message a member was found twice in the SET.

Action: Correct the value to include only one occurrence of each member in the SET.

343 BER_NOT_STRAIGHT_BER

Explanation: Encoding an ANY is impossible with only the information in the metadata. It might contain a value of literally any type - each of which would be encoded differently. The ANY type is deprecated and should not be used.

Action: Change the syntax to an ANY DEFINED BY if possible. Avoid the use of the syntax. If you must flow a value of this syntax, it must be provided in BER format. The BER format is an even number of hex digits surrounded by <>.

344 BER_UNRESOLVED_EXTERNAL

Explanation: An imported symbol was not found in the ASN.1 while processing a message. This will not be the case if CMIP services is initialized normally (all syntaxes are checked for completeness when the platform is initialized if they are contained in the normal set). If you added syntaxes to the user syntax section of the presentation initialization file, there may be unresolved externals. These will have been indicated by a warning message when the platform was started. If you have added syntaxes after initialization, these may not have been complete.

Action: Load additional syntaxes.

345 BER_STILL_MORE_STRING

Explanation: After parsing a message using the syntax information loaded, there was extra data in the buffer. The buffer must include exactly one syntactic construct - a complete message and no more. This will often happen if a string value includes too many closing parentheses at the end.

Action: If this is encountered in string processing, correct the string. If it is encountered while decoding a BER buffer, the syntaxes understood by the two systems is different. Align the syntaxes.

347 BER_DUP_MODULE

Explanation: The ASN.1 module you attempted to load is a duplicate of one already loaded. The name of the file that contained the duplicate module will be traced.

Action: Reload the ISTASN1 data set from the distribution media.

348 BER_UNRESOLVED_MODULE_REF

Explanation: While trying to resolve all of the imported symbols in the syntaxes loaded an entire module was not found. All references to it will be unresolved.

Action: Reload the ISTASN1 data set from the distribution media.

349 BER_UNRESOLVED_REF

Explanation: An external reference cannot be resolved in the ASN.1 syntax loaded. The module that was supposed to contain the type was found, but there was no such type name defined in the module. The list of all of the unresolved references will be written to the VTAM internal trace.

Action: The external reference that tried to use the type is likely wrong. Verify that you are trying to use a type that is defined in the module from which you are referencing it.

354 BER_FAILED_SUBTYPE

Explanation: The value provided was not allowed by the subtype specification.

Action: Change the value to be one of the value allowed by the subtype.

356 BER_BAD_CONSTRUCTED

Explanation: An element of the received BER buffer indicated in its tag that the value was a constructed value. The corresponding syntax loaded in CMIP services is a data type that cannot be constructed. These types are:

- INTEGER
- ENUMERATED
- BOOLEAN
- NULL

Action: Align the syntaxes in use by the peer systems.

357 BER_BAD_PRIMITIVE

Explanation: The value in the BER buffer for an explicit tag is encoded as a primitive type. It is not possible to have an explicit tag that is primitive since it must contain the other tag and a value.

Action: Align the syntaxes in use by the peer systems. Fix the peer system's encoding for explicit tags.

358 BER_BAD_INITIAL_OCTET

Explanation: The first octet of a BER buffer received from a peer application program is not correct. It does not represent a valid value for the data type being decoded.

Action: Align the syntaxes in use by the peer systems. It may be that the peer sent a message that is valid, but not within the scope of CMIP services.

359 BER_BAD_BOOLEAN

Explanation: A value received from a peer application program was being decoded as a BOOLEAN type. Its length was not 1 octet, which is required by BER.

Action: Align the syntaxes used by the peer application programs. Correct the encoding performed by the peer system.

360 BER_BAD_OI

Explanation: An OI value contained in a message being processed by the encode/decode component of CMIP services is not valid. If the message being processed is a string message from an application program the OI is not a legal dotted-decimal value. If the message is a BER buffer from a peer application program we have to trust the peer to have encoded a valid OI (it is only bits, after all). If this happens it will be preceded by one of two messages. One (PROGRAM_CHECK) indicates that the peer sent us an OI that will take more than 300 bytes to store in the string form. The other (MALLOC_ERROR) indicates we could not allocate memory.

Action: If the message was a string message, fix the value. If a MALLOC_ERROR happened, solve that problem. If you need to encode OIs that will be longer than 300 bytes in string form, call IBM Service.

361 BER_BAD_NULL

Explanation: A value for the type NULL contained in the BER buffer is not valid. The length is not zero. The peer system is not encoding values correctly or the syntaxes understood by the two systems are not the same.

Action: Align the syntaxes. If they are already aligned correct the peer application.

362 BER_EMPTY_BIT_STRING

Explanation: A received BER buffer contained a BIT STRING of length zero. The peer system is not encoding values correctly or the syntaxes understood by the two systems are not the same.

Action: Align the syntaxes. If they are already aligned correct the peer application program.

363 BER_BAD_PARAMETERS

Explanation: The encode/decode functions in CMIP services were called with an invalid parameter.

Action: Call IBM Service.

375 RDN_SEP_AT_BEGIN_OF_DN

Explanation: An RDN separator (;) was found at the beginning of a short-form DN.

Action: Correct the first character of the short-form DN.

376 AVA_SEP_AT_BEGIN_OF_DN

Explanation: An AVA separator (=) was found at the beginning of a short-form DN.

Action: Correct the first character of the short-form DN.

377 SPACE_AT_BEGIN_OF_DN

Explanation: A space was encountered at the beginning of a short-form DN. A short-form DN must begin with an OBJECT IDENTIFIER or a value reference (a label).

Action: Correct the first character of the short-form DN.

378 INVALID_CHAR_AT_BEGIN_OF_DN

Explanation: An invalid character was found at the beginning of a short-form DN. The first character of the short-form DN was not a digit, an alphabetic character, an RDN separator (;), an AVA separator (=), or a space.

Action: Correct the first character of the short-form DN.

379 RDN_SEP_AT_BEGIN_OF_RDN

Explanation: An RDN separator (;) was found at the beginning of an RDN.

Action: Correct the value of the short-form DN.

380 AVA_SEP_AT_BEGIN_OF_RDN

Explanation: An AVA separator (=) was found at the beginning of an RDN.

Action: Correct the value of the short-form DN.

381 SPACE_AT_BEGIN_OF_RDN

Explanation: A space was found at the beginning of an RDN while parsing a short-form DN.

Action: Correct the value of the short-form DN.

382 INVALID_CHAR_AT_BEGIN_OF_RDN

Explanation: There is an invalid character at the beginning of an RDN in a short-form DN.

Action: Correct the attribute type in the short-form DN.

383 INVALID_ALPHA_IN_INTEGER_VALUE

Explanation: An alphabetic character was found while processing an INTEGER form attribute type in a short-form DN.

Action: Correct the attribute type in the short-form DN.

384 INVALID_SPACE_IN_INTEGER_VALUE

Explanation: A space was found while processing an INTEGER form attribute type in a short-form DN.

Action: Correct the attribute type in the short-form DN.

385 INVALID_CHAR_IN_INTEGER_VALUE

Explanation: An invalid character was found while processing an INTEGER form attribute type in a short-form DN. This character was not an alphabetic character, a space or an AVA separator (an equals sign).

Action: Correct the attribute type in the short-form DN.

386 INVALID_SPACE_IN_OI_VALUE

Explanation: While parsing the attribute type in a short-form DN a space was encountered. The only valid characters are digits and period.

Action: Correct the attribute type in the short-form DN.

387 INVALID_CHAR_IN_OI_VALUE

Explanation: While parsing the attribute type in a short-form DN an invalid character was encountered. The only valid characters are digits and period.

Action: Correct the attribute type in the short-form DN.

388 INVALID_SPACE_IN_SYMBOLIC_VALUE

Explanation: While parsing an attribute type in a short-form DN, a symbolic value was found that contains a space. The attribute type must be either a valid OBJECT IDENTIFIER value (in dotted-decimal) or a symbol reference. The attribute type MUST be immediately followed by an equals sign.

Action: Correct the attribute type in the short-form DN.

389 INVALID_CHAR_IN_SYMBOLIC_VALUE

Explanation: While parsing an attribute type in a short-form DN, a symbolic value was found that contains characters other than letters and digits. It is possible that the attribute type was supposed to be an OBJECT IDENTIFIER, but its first character was a letter so it was interpreted as a symbolic value.

Action: Correct the attribute type in the short-form DN.

390 INVALID_CHAR_IN_ATTR_VALUE

Explanation: One of the two following errors occurred while parsing a short-form DN: a character other than the RDN separator (semi-colon) was found after close quote. A non-printable character was found in a value.

Action: Correct the value for the short-form DN.

391 INVALID_SPACE_IN_ATTR_VALUE

Explanation: A value portion of an RDN in the short-form DN contained a space and the value was not surrounded by quotes. This is ambiguous; the platform does not know whether the space is part of the value, or merely white space.

Action: Correct the value of the short-form DN by eliminating the space or surrounding the value in quotes.

392 PREMATURE_END_OF_DN

Explanation: A short-form DN value was incompletely specified. A short-form DN must be composed of complete RDNs. Each RDN must include a type (OI), an equals sign and a value. This may have occurred due to a dangling semicolon at the end of the name, or it may be due to an RDN with a type but no value.

Action: Correct the value of the short-form name.

393 INVALID_SPACE_AT_END_OF_RDN

Explanation: While parsing a short-form DN, CMIP services found a space immediately following the RDN separator (the semi-colon). This is not allowed. This must be the beginning of the next object identifier and object identifiers cannot contain spaces.

Action: Correct the name value by deleting any spaces in the OI portions of all RDNs.

394 BOTH_QUOTE_TYPES_USED

Explanation: A short-form DN value was contained both kinds of quotes and the attempt to surround it with quotes (during tranformation to long-form) failed.

Action: Correct the value of the short-form name.

400 REPL_ERR_INVLD_VERBCODE

Explanation: This indicates that a message is being passed through CMIP services which is not of a known type. In general, CMIP services expects CMIP messages or a small handful of internal utility messages. The message that CMIP services just received was not one of the types that the code can process.

Action: Call IBM Service.

401 REPL_ERR_MISSING_ASN1_TREE

Explanation: This indicates that a message was received that was basically valid (i.e. CMIP services recognized the message type and a couple of header fields), but the ASN.1 parse tree (which is required for all processing) was missing from the message. Processing on this message cannot continue.

Action: Call IBM Service.

402 REPL_ERR_OBJCLASS_MISSING

Explanation: The object class is a required field in virtually all CMIP messages. If the CMIP message that CMIP services is processing requires an object class and one is not present, this error will be logged. The fact that a required attribute is missing should have been noticed prior to CMIP services receiving the message, therefore this indicates an internal error.

Action: Call IBM Service.

403 REPL_ERR_OBJCLASS_INVALID

Explanation: The value of the managed object class component in the message is not recognized as a valid value. This can either mean that the valid GDMO definition has not been loaded by CMIP services, or a truly invalid value was specified.

Action: Correct the object class.

404 REPL_ERR_OBJINST_MISSING

Explanation: This indicates that either a CMIP message that requires the managed object instance component did not have one, or (more likely) that a locally generated request that requires the managed object instance attribute did not specify one.

Action: Include a valid managed object instance in the request.

405 REPL_ERR_OBJINST_INVALID

Explanation: This indicates that the specified managed object instance is in an invalid format and could not be encoded. There are a number of cases where this can occur:
- During CMIP message processing
- During processing of a locally generated request

Action: Verify the specified managed object instance against the associated naming rules in the name bindings used to construct the name.

406 REPL_ERR_DUPLICATE_OBJINST

Explanation: This indicates that either a CMIP message was received which tried to create an instance that already exists, or a local registration was attempted for an instance that already exists.

407 REPL_ERR_NO_SUCH_OBJINST

Explanation: This indicates that the managed object instance specified in the CMIP message or local request could not be found in the current instance tree. This could mean that some of this instances parents were not present in the tree either.

408 REPL_ERR_MOI_OC_MISMATCH

Explanation: During the processing of a non-create CMIP message CMIP services determined that the managed object class specified in the message was not the actual class object identifier (2.9.3.4.3.42), nor the actual managed object class of the specified instance, nor one of the allomorphic object classes of the specified instance.

Action: Change the original CMIP request to either 2.9.3.4.3.42 or to a correct managed object class for the specified managed object instance.

409 REPL_ERR_NAME_CREATE_FAILED

Explanation: An error occurred creating an object instance.

Action: Look at CMER records in the VTAM internal trace for additional information.

410 REPL_ERR_GDMO_FILE_BAD_VERS

Explanation: This indicates that CMIP services attempted to load an initialization file with a version number different from the version currently implemented in CMIP services.

Action: Reload the ISTGDMO data set from the distribution media.

411 REPL_ERR_NOTHING_TO_DELETE

Explanation: This indicates that the delete was directed at a specific managed object, but that managed object (which exists) cannot be deleted for some reason. The main reasons for this are all related to name binding rules. The managed object instance may not be deleteable, or it might only be deleteable if it contains no instances (and does contain instances), or it should delete contained instances but one or more of them is not deleteable.

Action: Verify the name binding rules for the managed object instance that was to be deleted to determine why the instance could not be deleted. You might need to specify a scope in order to delete the whole sub-tree. Perhaps you shouldn't be attempting to delete this instance at all.

412 REPL_WRN_OBJCLASS

Explanation: This indicates that one or more of the object classes that were specified in the list of allomorphs or create handlers on a local registration are either unknown (invalid) or do not allow creates (for create handler list).

Action: During run time the application program should determine if the object classes that were rejected are a problem. If so, the object should probably be deleted. If it is not a problem, nothing needs to be done since the object was registered without the erroneous classes. For future runs the application program code should be fixed to use a valid set of managed object classes. This means that the managed object class object identifiers should all be loaded in the CMIP services initialization file, and that all of the managed object classes specified in the create handlers list should be createable.

413 REPL_ERR_ALREADY_AN_STM

Explanation: This indicates that during the final phase of registration for a new instance, which was requesting to be a subtree manager, a parent instance was found which was already a subtree manager. Nested subtree managers are not legal.

Action: You should check to see if the parental subtree manager is one you expected to be there. If it is, then you either need to move the new subtree manager to a new location (or don't register it as a subtree manager). If the existing subtree manager is not supposed to be there, try to figure out how it got there and get rid of it.

414 REPL_ERR_INVLD_STM_CHILD

Explanation: Once a subtree manager has registered control of a portion of the naming tree, only instances registered over the same application program connection are allowed on that subtree. If an instance from another application program connection tries to register under a subtree reserved by a different application, the new registration will be refused.

Action: Do not attempt to register a new instance under some other application program's subtree.

415 REPL_ERR_SCOPES_TO_NOTHING

Explanation: This indicates that there was a scoped message which could have addressed multiple instances but in the end addressed none. The list could be pared down due to name binding rules (for deletes), or access control.

Action: If there were instances that you wanted this to be sent to, consider altering the scope to try to include the instances. Perhaps you should re-evaluate to determine if you are actually able to address the instances that you were trying to.

416 REPL_ERR_INVALID_SCOPE

Explanation: This indicates that either the scope is syntactically incorrect, or that the destination was GlobalRoot (i.e. a NULL distinguished name) and the scope included level 0 (level zero does not exist for GlobalRoot scoping).

Action: Compare your scope to the standards. Verify that it does not include level 0 for a GlobalRoot scope and that it is syntactically correct according to the standards.

417 REPL_ERR_COMMITDN_NOTIN_LIST

Explanation: This indicates that an attempt was made to process an instance which was thought to be pending registration (either to complete the registration, or to terminate the registration). This instance was not found on the list of pending creations.

Action: There is no recovery action for this. It indicates that either invalid instance information was passed in, or that the pending instance was removed during the cleanup processing of a related instance.

419 REPL_ERR_NO_ONE_2_SEND_CRT_2

Explanation: A create was received for a managed object class which does not have a registered create handler for it. A create handler is an instance that indicates that it is capable of receiving, processing, and responding to CMIP create messages for a specified managed object class. If there is no create handler registered for a specified managed object class, CMIP services does not know where to send the create for processing.

Action: If you do not wish to handle creations for the specified managed object class, then nothing needs to be done. If you would like to be able to accept creates for the specified managed object class, then an application program must register an instance with CMIP services as a create handler for the specified managed object class.

420 REPL_ERR_NOONE_2_SEND_EVNT_2

Explanation: An event report or notification was received which had no specific destination associated with it (an unambiguous AE title) and there was no event handler registered with CMIP services.

Action: Call IBM Service.

421 REPL_ERR_ALREADY_EVNT_HNDLR

Explanation: An attempt was made to register an instance with the event handler capability set, but there is already an event handler registered. CMIP services only supports the existence of one event handler at a time.

Action: Call IBM Service.

422 REPL_ERR_NAMEBIND_INVALID

Explanation: This indicates that the name binding that was specified was either in an invalid format (primarily this means length 0), or that the value specified could not be found in the tables of valid name bindings. The tables are loaded by CMIP services at initialization time.

Action: Check to make sure that a valid name binding value was specified.

424 REPL_ERR_CRT_FAIL_NB

Explanation: Either the name binding that was specified does not allow creation via CMIP create messages, or there was no name binding specified and one could not be found that allowed CMIP create messages.

Action: Either specify a different name binding (one that allows CMIP creates), or rethink whether you should be trying to remotely create an instance of this managed object class.

425 REPL_ERR_CRT_FAIL_NO_NB

Explanation: This indicates that there is either no name binding to create an instance of the specified managed object class under the managed object class of the specified parent instance, or there is no legal name binding which has a naming attribute that matches the requested naming attribute of the new instance.

Action: First check that there is a name binding that uses the desired naming attribute, and verify that this name binding is being loaded in CMIP services initialization file. Second check that the desired name binding allows creation of an instance of the specified managed object class under the managed object class of the parent. If it doesn't, consider picking a different parent, a different managed object class for the new instance, a different name binding (or perhaps specify a name binding if you were not), or change the attribute type of the final RDN to one that matches a useful name binding.

426 REPL_ERR_DLT_FAIL_CONTOBJS

Explanation: This indicates that the base instance that this delete was sent to (either no scope, or a base only scope) only allows deletes if the instance does not contain any child instances - and the instance does contain child instances.

Action: Either delete all of the kids specifically, or include a scope with the delete in order to wipe out all instances at and below the base instance.

427 REPL_ERR_DLT_FAIL_TO_DCO

Explanation: This indicates that some instance inside of the sub tree of the base instance cannot be deleted (even though the base instance's name binding indicates that it should delete contained instances). This could be because of access control, or because the name binding of the child instance does not allow deletes in some way (no deletes at all, only if no contained objects and it contains objects).

Action: Try deleting the child instances individually, or with a scope that includes the whole subtree.

428 REPL_ERR_DLT_FAIL_NB

Explanation: This indicates that the instance is not allowed to be deleted by use of CMIP delete messages. The name binding indicates that this instance cannot be deleted.

Action: You cannot delete the specified instance. Perhaps the instance should not deleted, or perhaps it should have been created with a different name binding.

429 REPL_ERR_NO_LOCALDN

Explanation: A CMIP message was sent with the local DN form of distinguished name specified, but there is no instance registered as a local DN handler for the AE title of the association that this message was received over. Local instances can register with CMIP services indicating that their distinguished name can be used as the initial RDNs for any local DN form message received over an association with a specified AE title.

Action: Either a local DN handler should be registered for the desired AE title, or the CMIP message should not use local DN form of distinguished name.

430 REPL_ERR_DUPLICATE_LDNH

Explanation: An instance tried to register as a local DN handler for an AE-Title that already had another instance registered as a local DN handler for it. CMIP services does not allow multiple Local DN handlers to register for the same AE-Title since there is no way to determine which one to choose.

Action: Determine which of the instances should be the Local DN handler and register it first (or only register that valid one). If you don't care about receiving the error message, but want to make sure that there is a Local DN handler for the AE-Title, then by all means, make as many registration attempts as you want.

431 REPL_REG_CREATED

Explanation: This is an internal error code. It should never be externalized.

Action: If this error is logged or externalized, a programming error exists in CMIP services. Call IBM Service.

432 REPL_CRT_COMPLETED

Explanation: This is an internal error code. It should never be externalized.

Action: If this error is logged or externalized, a programming error exists in CMIP services. Call IBM Service.

433 REPL_REG_COMPLETED

Explanation: This is an internal error code. It should never be externalized.

Action: If this error is logged or externalized, a programming error exists in CMIP services. Call IBM Service.

434 REPL_REG_SUSPENDED

Explanation: This is an internal error code. It should never be externalized.

Action: If this error is logged or externalized, a programming error exists in CMIP services. Call IBM Service.

435 REPL_ERR_ATTRTYPE_MISMATCH

Explanation: The attribute type of the final RDN did not match the object identifier of the naming attribute for the specified name binding. If you specify a name binding and a full distinguished name (including the new final RDN) CMIP services checks to make sure that they are internally consistent. Receiving this error indicates that you provided inconsistent values.

Action: Provide a consistent name binding and distinguished name.

436 REPL_ERR_CANNOT_CHANGE_NB

Explanation: During the first phase of processing for CMIP create requests, a name binding is either specified or selected for any non-auto instance naming forms of creates. The name binding is specified or chosen based on a number of factors including the validity of the naming attribute and the name binding's ability to be created. If this value is changed in the second phase of create processing to something that changes some of these values (such as changing the naming attribute), the values may not be legal any more. It is illegal to select a name binding that would invalidate the instance creation information.

Action: If you must change the name binding value, select a value that uses the same naming attribute, allows instance creation, and allows the same managed object class to be instantiated under the same parent managed object class.

439 REPL_ERR_NB_DISALLOWS_NEWOC

Explanation: There are three points to this triangle of validation. Two of these points are fixed. First is the managed object class of the parent (which is fixed). The second is the name binding (which is also fixed). The third is the managed object class of the instance being registered (this is what you just tried to change from the original value). The new managed object class must allow the use of the original name binding to create a new instance of the new managed object class under an instance of the parents managed object class.

Action: Either use the original managed object class, or pick a managed object class that allows the use of the old name binding under the existing parent instance.

---

440 REPL_ERR_SYNC_NOT_SUPPORTED

Explanation: Atomic synchronization is not currently supported by CMIP services. Atomic synchronization is rejected if that option is specified and the scope of the message includes any children of the baseManagedObjectInstance. The default for the synchronization field is bestEffort.

Action: Specify bestEffort, remove the synchronization field and allow it to default, or trim the scope to include only the baseManagedObjectInstance.

---

500 CRC_ERR_INVLD_VERBCODE

Explanation: This indicates that a message is being passed through CMIP services which is not of a known type. In general, CMIP services expects CMIP messages or a small handful of internal utility messages. The message that the CMIP component just received was not one of the types that the code can process.

Action: There is not much that can be done about this. Primarily this means that there was an internal error of some kind that should be reported back to IBM. Please note all of the information that is logged along with this error code (as well as any logs immediately before and after this one). CMIP services will attempt to reject this message and continue processing.

---

501 CRC_ERR_INVLD_SESSHAND

Explanation: This indicates that somehow the CMIP component received a message that contained a Association or Session handle which was not a valid value. In most cases this indicates an aborted session, but application program s are allowed to specify the Association/Session to use for routing the message. This error might be reported because an invalid value was specified by the application. It is rare that this error is caught in the CMIP component since the messages pass through other components which validate these fields before it gets to the CMIP component.

Action: If the application program selected a bad value, the application program should be fixed. If the application program s value was valid or the application program did not specify a value, then it is likely that the Association/Session was aborted. CMIP services will reject this message and continue processing.

---

502 CRC_ERR_INVLD_INVOKEID

Explanation: This means one of the following:

- A CMIP CancelGet was attempted using an invoke id that was not for a get request.
- The invoke id field was missing from the message.
- The invoke id on a response/confirm does not match any of the outstanding indication/request invoke ids.

There are two primary causes of this set of problems:

- The application program specified a bad invoke id value.
- The Association/Session over which this message was traveling has been aborted.

Action: Determine if this is an application program error (if the application program returns a different invoke id value than was passed to it in the original message). Fix the application program error if that's what it was. If the invoke id value was valid, check to see if there were any error messages logged to indicate that the Association/Session was aborted. If it was, determine if there is any action you can take to prevent it from happening again. If there was no error logged, then the Association/Session was aborted in a "normal way". CMIP services will reject the message and continue processing.

---

503 CRC_ERR_DPLCT_INVOKEID

Explanation: This indicates that an application program (local or remote) tried to re-use an invoke id that has not yet been completed. CMIP services does not time out invoke ids or re-use invoke ids in any way. But an application program can specify its own invoke ids, or the remote application program may be using a stack other than CMIP services which does time out and re-use invoke ids. In this event an outstanding invoke id, which has not yet completed processing may be re-used by an application. This is an error. The CMIP standards do not provide a way for invoke ids to be timed out therefore CMIP services does not time them out. The most likely cause of this problem

is that the local application program is either taking to long to process the message or has an error, and the remote requester times out the invoke id then tries to re-use it.

Action: First you should check why the application program might be taking too long to answer. Fix this if you can. Second you can try to extend the timeout length of the remote stack, or eliminate timeouts all together. CMIP services will reject this message and continue processing.

---

504 CRC_ERR_INVLD_LINKEDID

Explanation: This indicates that an attempt was made to send a linked reply using an invoke id value (invoke id of the original request/indication that is) in the response that is not an outstanding invoke id. In other words the application program is attempting to send linked replies to an unknown request. The most likely cause of this is that the Association/Session was aborted and the messages on that Association/Session were cleaned up. It is also possible that the application program filled in the wrong linked id value in the linked response. Note that the linked id being passed back in the linked reply should be the same value as the invoke id of the original message.

Action: Determine if this is an application program error (if the application program returns a different invoke id value than was passed to it in the original message). Fix the application program error if that's what it was. If the invoke id value was valid, check to see if there were any error messages logged to indicate that the Association/Session was aborted. If it was, determine if there is any action you can take to prevent it from happening again. If there was no error logged, then the Association/Session was aborted in a ″normal way″. CMIP services will reject the message and continue processing.

---

505 CRC_ERR_UNABLE_TO_BUILD_MSG

Explanation: This indicates that the CMIP component is attempting to construct the final full message (which may be a reject or error response) to pass on to the next stage of processing, but is unable to complete the construction of the message. The most likely cause of this is an out of memory condition, but it could also be related to an internal error. An out of memory condition will be logged in a separate error log message from the component that discovered it.

Action: Refer to other errors in the trace.

---

506 CRC_ERR_INVLD_ROERRJ_RCVD

Explanation: This can occur when an application program goes away (primarily a manager application), or an Association/Session is aborted and one of the partners doesn't realize it yet. The response is sent and the invoke id is cleaned up locally, then the partner rejects the message back because the partner is not present. When the invoke id is looked up, it cannot be found. The condition is logged and the message is ignored (no further processing is possible). CMIP services will continue processing.

Action: There is no action that can or needs to be taken for this.

---

507 CRC_ERR_INVLD_CANCELGET

Explanation: The application program issued a CancelGet request for an outstanding invoke id. The outstanding invoke id was located, however, it was not a Get Verb. This is a violation of the CMISE standard. For a CancelGet request, the request is rejected back to the application. For a CancelGet indication, an ROER is sent back to the application program which generated the CancelGet.

Action: Ensure that the application program responsible for generating the CancelGet request is correctly inserting the invoke id of the Get to be canceled into the CancelGet.

---

508 CRC_ERR_INVLD_INVOKEID_ON_CANCELGET

Explanation: The application program issued a CancelGet request for an invoke id that could not be located by CMISE. For CancelGet requests, the request is rejected back to the application. For CancelGet indications, an ROER is sent back to the application program which generated the CancelGet.

Action: The original Get request may have completed processing before the CancelGet processing had begun.

---

509 CRC_DELETE_RORJ_RECEIVED

Explanation: Because delete operations cannot be backed out, by the time the CMISE protocol machine receives an RORJ from the peer protocol machine indicating that the delete response is invalid it is too late to terminate the delete. So the RORJ is ignored and this message is logged as a warning.

Action: Determine why the peer entity rejected the delete response. No action is required on CMIP services system logging the warning.

550 SSERR_STATE_INVALID

Explanation: A message was received in the Session Layer that implies a violation of the Session Layer protocol. This may be any verb received out-of-sequence on an established connection, or any verb other than an S-Connect received when the identified session does not exist. This will only occur if the peer entity does not implement its Session protocols correctly, or during the short period while a session is being torn down abruptly. If all Sessions/Associations are terminated with the graceful Release protocols, this will never occur. When one side of a communication dies, it is possible that one or more messages will flow from the upper layers of CMIP services before this is noticed. This will cause the messages that cause this error to be logged to be lost, just as they will be after the upper layers are notified, since the session/association they need to use is no longer in existence.

Action: No action is required.

551 SSERR_SPDU_INVALID

Explanation: A received SPDU (Session Protocol Data Unit) was invalid. It was not correctly formed according to the rules for Session Layer headers and data. The peer entity is producing bad SPDUs or the underlying Transport layer (which is supposed to provide a reliable packet delivery service) has corrupted the message.

Action: This really should never happen. We have never seen it happen with any of the partner implementations. If it does happen, that data stream will need to be analyzed to determine whether the messages (specifically this one and the message before and after it) conform to the definition of SPDUs. If they do not, the sender or underlying transport need to be analyzed to determine the cause. If they do, a trace of the traffic for this connection should be sent to IBM Service.

552 SSERR_MISSING_PI

Explanation: A mandatory piece of information was omitted from an SPDU. This was either the reason on an S-Refuse or the Transport disconnect on an S-Abort. In either case the connection will be terminated (as it would have been if the flows had been correctly formatted). This will be logged to indicate that the peer entity is not conforming to the defined protocol.

Action: Check the S-Refuse and S-Abort messages produced by the peer. Correct them to conform to the protocol. The overall result will be correct in either case - a session that should have been terminated will be terminated.

553 SSERR_MISSING_UI

Explanation: An S-Data indication was received from a peer entity that included no data. Since this is an error (and a waste of the network to be sending packets that contain nothing) the connection will be closed.

Action: Correct the peer application.

554 SSERR_VERB_INVALID

Explanation: A message received by the Session Layer from another layer in CMIP services was invalid. This is an internal error in CMIP services.

Action: Collect all log information and any re-creation scenario possible and call IBM Service.

581 SSERR_DUPLICATE

Explanation: A duplicate session identifier has been assigned by the Session Layer. This is an internal error in the platform.

Action: Call IBM Service.

557 SSERR_USERDATA_SIZE

Explanation: The protocol and profiles specify limits on the size of the user data included in the various Session Layer PDUs. One of these limits has been violated. Normally this is the limit specified the profile of 10240 octets of data on an S-Data message.

Action: Decrease the size of the data provided in a single message.

559 SSERR_TDISC_CONGESTED

Explanation: There was insufficient storage available to transmit the data on the connection. Congestion has occurred locally or remotely. In an attempt to relieve this congestion this message will be discarded and the connection terminated.

Action: Eliminate some of the traffic between these two systems or increase the resources allocated to communication between the two systems. This is often a transient error, and merely re-establishing the connection will work.

560 SSERR_TDISC_UNATTACHED

Explanation: A connection could not be established with the peer system. Either the system is not running, or the platform is not running on the system.

Action: Check the address included in the log for this error. If it is correct check for connectivity with the system and make sure a platform is running on the system. If is is incorrect you need to determine why the address was chosen. It is derived from the destination information contained in the original message. There is a two step mapping performed - mapping instance name to AE-Title and AE-Title to address. The original name may be incorrect, the mapping from name to AE-Title may have produced an unexpected AE-Title, or the mapping from the AE-Title to address may have produced an unexpected result.

561 SSERR_TDISC_ADDRESS

Explanation: The remote address is not recognized for network routing. Either the address is incorrect, or the system is not running.

Action: Check the address included in the log for this error. If it is correct check for connectivity with the system. If is is incorrect you need to determine why the address was chosen. It is derived from the destination information contained in the original message. There is a two step mapping performed - mapping instance name to AE-Title and AE-Title to address. The original name may be incorrect, the mapping from name to AE-Title may have produced an unexpected AE-Title, or the mapping from the AE-Title to address may have produced an unexpected result. See the description of the naming service and directory mappings to correct this.

562 SSERR_VERSION

Explanation: The session version indicator received on an S-Accept is not version 2. Only version 2 is supported. This is the version specified in the profiles for management systems.

Action: Correct the peer application program to implement or use version 2 protocols for the Session Layer.

563 SSERR_PARTNER_ABORT

Explanation: An abort was received from the peer entity. The session is being torn down. This may represent normal operation - if the partner issued an abort. This condition is logged to allow problem determination to know that CMIP services received an S-Abort from the peer system.

564 SSERR_ENCLOSURE_ITEM

Explanation: The enclosure item was found in an SPDU, but segmenting is not supported. This should not happen - we negotiate away segmentation.

Action: Correct the peer application program to eliminate segmentation.

568 MD_ERR_BAD_MDSMU

Explanation: A badly formed MDS-MU was received. The single place that traces this error will also trace the sense code (what was wrong with the MDSMU) and the entire MDSMU trace.

Action: Fix the message sent by the partner application.

573 MD_ERR_SNACR_BEING_SENT

Explanation: An SNA condition report is being sent to the partner application program indicating an error has occurred. The log will include the sense code of the error and the SNA condition report (SNACR) being sent to the partner.

Action: Correct the condition indicated by the sense code.

574 MD_ERR_SNACR_RECEIVED

Explanation: MDS interface received a SNACR. The sense code and SNACR are included in the log.

Action: Analyze the sense code and SNACR to determine what error has occurred in the underlying SNA transport.

578 SSERR_GIVE_TOKEN_NO_DATA

Explanation: A Session Give Token PDU was received with no data following it. This is invalid for the kernel of the Session Layer.

Action: Correct the peer application.

802 ACF_EVENT_LOOP

Explanation: Sending the event to the indicated AE-Title would cause an infinite loop in event processing. It is not being sent. This event report attempted to use an association that is local. This would cause the event to be routed back to CMIP services for processing causing an infinite loop. The destination of an EFD must contain either the name of a local instance or the AE-Title of a remote AE.

Action: Change the destination on the EFD to represent a local instance or a remote AE.

803 ACF_INVALID_ASSOC_ID

Explanation: A message was received by CMIP services from one of its application programs. This message included an association identifier (either because it was a response or because the initiator wished to use a specific association for the request). The association identifier does not represent a currently active association, so the message cannot be sent. This may have occurred because:

- The application program used an invalid handle that never represented a valid association.
- The application program is trying to use the same association for all of its requests and the association has been terminated.
- The application program is attempting to respond to an indication and the association terminated between the time the indication was received and the response was sent.

Action: For responses, use the correct association handle, exactly the information provided in the source of the indication. For requests use normal routing (do not include an association handle) or correct the handle value.

806 ACF_INVALID_USER_ID

Explanation: An object asked to terminate an association which was started by another object explicitly. This is not allowed. When an object asks to start an association using the ACF.Associate message and that object has registered as an AE, the association is reserved for its use. No other object will be allowed to use it or terminate it.

Action: Send the termination request from the correct object.

807 ACF_RSP_BUILD_SEND_FAILED

Explanation: A message could not be built because the ASN.1 data sets are incorrect.

Action: Reload the ISTASN1 data set from the distribution media.

808 ACF_ERR_KILL_LOC_ASSOC

Explanation: The association is used to provide local (logical) connectivity to local Application Entities. They are automatically established and terminated when Application entities are registered and terminated. These associations do not represent any real network resources, so there is no reason to terminate them.

Action: Do not try to terminate these associations.

812 ACF_BAD_AE_TITLE_FORMAT

Explanation: The AE-Title provided as the value for CMIP services could not be encoded or processed by the current set of ASN.1 definitions.

Action: Reload ISTASN1 from the distribution media.

814 ACF_CANNOT_FIND_INST

Explanation: The instance name provided with this message (in the baseManagedObjectInstance field of most CMIP requests) cannot be resolved into a potential serving AE-Title, so CMIP services does not know where to send the message.

Action: Possible actions include:

- Correct the instance name if it is incorrect.
- Add an entry to define a mapping for this name to AE-Title in the directory definition file and restart the platform.
- Add an explicit AE-Title as the destination of the request.
- Add an explicit association handle as the destination of the request.

815 ACF_NO_DESTINATION

Explanation: This error will be returned if a message does not include any of the following types of destination information:

An association handle

An AE-Title

A DistinguishedName

An instance name in the CMIP message

The only way this should be possible is if the request is sent to 'GlobalRoot' (an instance name with no RDNs) and no other information is provided to direct the message to the correct system.

Action: Put some type of destination information in the message. If this CMIP message is trying to use 'GlobalRoot', you must provide one of the other types of destination. This is the only case where additional destination information is required.

817 ACF_NO_ASSOC_TEMP

Explanation: The required association could not be established. There are several possible causes for this, each of which will cause additional errors to be logged. The causes include:

There is no CMIP platform running on the designated target machine.

The address does not represent a real machine at all.

There is a CMIP platform running but its capabilities do not match those in use by this platform.

Action: Look for other trace entries to determine the real cause of the error.

819 ACF_EMPTY_DEF_LIST_RESULT

Explanation: While trying to negotiate a common set of syntaxes with a potential peer system we discovered that the two systems have NO syntaxes in common. Since this will not result in any communication, we will not establish the association. This should not ever happen if we are actually trying to connect to another system that implements CMIP. It could happen if we mistakenly try to connect to an implementation of X.500 or X.400, so we really do not want such an association to be established.

823 ACF_QUEUED_MESSAGE

Explanation: This should not occur in an error message. It may occur in a trace. This is the normal mode of operation when a new association needs to be established.

824 ACF_ASSOC_ID_WRAP

Explanation: The identifiers assigned to associations have just wrapped. Unpredictable behavior may occur if there are collisions. Collisions are extremely unlikely since they are assigned sequentially from a 32bit space. If they do collide the platform will begin to route messages incorrectly.

Action: Restart CMIP services.

826 ACF_TOO_MANY_LOCAL_ASSOCS

Explanation: Local associations are limited to 100 at any given time.

Action: Possible actions:
- Terminate a local association.
- Terminate a local AE.

827 ACF_DUPLICATE_AE

Explanation: An attempt was made to register a local AE-Title for an object instance. This AE-Title is already in use by another instance on this system.

Action: Change the AE-Title or terminate the previous object using this AE-Title.

828 ACF_REMOTE_AE

Explanation: An application program attempted to register a local AE-Title that is identical to the AE-Title currently being used by a remote Application Entity.

Action: Choose a different AE-Title, or terminate the associations with the remote entity (and make sure it never re-connects using the same AE-Title).

829 ACF_INVALID_STATE_TO_RELEASE

Explanation: An application program attempted to cause CMIP services to Release an association. When the association was checked it was determined that it was not in the associated state. A Release message can only be sent when an association is in the ASSOCIATED state without causing a protocol violation.

Action: If you really want to terminate the association, use an Abort instead of Release.

830 ACF_INVALID_AE

Explanation: An instance was attempting to register itself as the local DN handler for all messages that are received on associations to a specific AE-Title. The AE-Title it specified is not one of those currently supported by the local system, so it will never be used.

Action: Choose the correct AE-Title (possibly &AET) or register the AE-Title and retry the request.

831 ACF_BAD_P_MODE

Explanation: Only normal mode Presentation layer protocols are supported by CMIP services. The peer entity tried to establish a connection using some other mode.

Action: Change the peer to use normal mode Presentation Layer protocols.

832 ACF_BAD_P_PROTOCOL_VERSION

Explanation: Only version 1 Presentation layer protocols are supported by CMIP services. The peer entity tried to establish a connection using some other version.

Action: Change the peer to use version 1 Presentation Layer protocols.

833 ACF_BAD_CMIP_VERSION

Explanation: This implementation only supports version 2 of the CMIP protocol.

Action: Use CMIP version 2 for management flows. Change the peer to negotiate version 2 of CMIP.

834 ACF_BAD_APPL_CONTEXT

Explanation: CMIP services supports a specific set of application program contexts to assure the platform that the peer is actually talking the same language. The supported contexts are:

ISO

CCITT

NM Forum

Action: Try to connect from CMIP services to the peer - maybe it will accept the ISO context (or the appropriate CMOT context if using CMOT). Adapt the peer to support one of these protocols. Support for additional contexts should not be necessary - these are all of the common contexts for OSI management. If additional contexts are necessary, contact IBM Service

835 ACF_NO_APPL_CONTEXT

Explanation: The A-Associate indication received form a peer Application Entity did not include any application program context. This does not allow us to confirm that it is actually using CMIP, or even how to resolve the details of the A-Associate indication. This association will be rejected.

Action: Establish the association to the peer (maybe it will accept our context) or adapt the peer to send an application program context CMIP services supports.

836 ACF_NO_APPL_INFO

Explanation: The platform received a P-Connect-Indication that contained no application program s layer information. There was NO A-Associate-Indication contained in the PDU. Since this would not result in a usable connection, the connection will be rejected.

Action: Establish the association to the peer (maybe it will accept our A-Associate) or adapt the peer to send an A-Associate on the P-Connect.

838 ACF_WRONG_AE_TITLE

Explanation: The A-Associate-Indication provided a value for the called-AP-Title and qualifier that does not match the local values. The A-Associate could be rejected, but it will be accepted. We will merely respond with the local AE information in the responding AP-Title and responding AE-qualifier. The peer can abort the association if it sees fit.

Action: None - the association was established.

840 ACF_NO_AE_QUALIFIER

Explanation: This indicates (internally) that an A-Associate-Indication was received that contained only an AP-Title, and no AE-Qualifier. The association will be accepted.

Action: No action is required unless this error code is externalized.

900 MB_ERR_PROCFAIL_NOT_OPTIONAL

Explanation: Although the parameter for an ROER-processingFailure is specified as OPTIONAL in the CMIP standard, the argument for an ROIV-m-Linked-Reply is not OPTIONAL. Because CMIP services may need to reformat an application's processingFailure CMIP error from an ROER to an ROIV-m-Linked-Reply, CMIP services requires the processingFailure argument to be specified.

Action: Correct the application program to specify a processingFailure argument in all cases. The following is an example of a processingFailure argument that specifies the genericSpecificError: ″(&OC, (distinguishedName &DN), (1.2.124.360501.9.24, NULL))″

901 MB_ERR_COMPXLIM_NOT_OPTIONAL

Explanation: Although the parameter for an ROER-complexityLimitation is specified as OPTIONAL in the CMIP standard, the argument for an ROIV-m-Linked-Reply is not OPTIONAL. Because CMIP services may need to reformat an application's complexityLimitation CMIP error from an ROER to an ROIV-m-Linked-Reply, CMIP services requires the complexityLimitation argument to be specified.

Action: Correct the application program to specify a complexityLimitation argument in all cases. The following is a minimal example of a complexityLimitation argument which leaves out all the optional members: ″()″

903 MB_ERR_INVALID_TYPENAME

Explanation: An ASN.1 type name was not recognized.

Action: Correct the type name.

904 MB_ERR_NOT_CONNECTED

Explanation: A message was received from an application program that is no longer connected.

Action: Call the MIBConnect function.

914 MB_ERR_DELETE_PROTOCOL_ERROR

Explanation: Various rules limit the responses which an agent is allowed to make in the first phase of a CMIP delete, when it send the MIB.DeleteResponse syntax to CMIP services. This error, returned to the agent application, indicates that the response was not allowed. The reason depends on whether the instance is a subtree manager or not and where the instance falls within the scope of the delete. This error is returned for six distinct conditions:

1. The instance is a subtree manager and is above the scope of the delete and has answered accepted (0).
2. The instance is a subtree manager and is above the scope of the delete and has answered rejected (1).
3. The instance is a subtree manager and is below the scope of the delete and has answered stmChildrenOnly (2).
4. The instance is not a subtree manager and has answered stmChildrenOnly (2).
5. The instance is below the scope of the delete and has answered noOneSelected (3).
6. The instance has not answered with 0, 1, 2, or 3.

Action: Correct the application program 's delete-handling code. Note that if the object instance is not a subtree

manager (the normal case), then conditions 1-3 are eliminated as possible causes. Also note that condition 4 does not specify where the instance is relative to the scope of the delete because non-subtree manager instances are never allowed to answer stmChildrenOnly. Condition 5 is an error because the filter for the delete is always stripped from the delete indication before it is delivered to the instances which are below the scope of the delete. Because these instances did not receive the filter, they cannot possibly have failed to pass it.

---

918 MB_ERR_INVALID_LINK_ID

Explanation: The value specified on the link identifier parameter does not refer to a valid connection.

---

919 MB_ERR_INVALID_STATE

Explanation: CMIP services was attempting to write a message to a client application program but determined that the connection was not in a useable state. The message was not written, and the error was logged. The application program program was not notified of the error nor was the sender of the request.

Action: The application program should exit and reinitialize.

---

920 MB_ERR_NOT_REGISTERED

Explanation: The application program has indicated that it has had an unrecoverable error when returning to the read queue exit routine or the data space is out of storage. The registration will not be allowed.

---

921 MB_ERR_CMIP_ERR_RESP_ILLEGAL

Explanation: An agent application program attempted to return a CMIP error and CMIP services flagged the response as illegal because the error code specified in the response is not allowed for indications of the type being responded to. For example, if an client agent application program returns a getListError for an m-Set indication or an invalidArgumentValue error for a an m-Create. This error is also returned by CMIP services when an agent responds to a delete with the syntax MIB.DeleteResponse(1, X, ...) where the X is an error code that is not allowed by the CMIP standard in response to an m-Delete indication.

Action: Correct the client agent application program to return an error code compliant with the CMIP standard (Rec. X.711 | ISO/IEC 9596-1 second edition).

---

922 MB_ERR_CMIP_ERR_RESP_STKCHK

Explanation: An agent application program attempted to return a CMIP error and CMIP services flagged the response as illegal because the error code specified in the response was checked for by CMIP services when the indication was processed and was verified at that time not to have occurred. For example, all object classes are looked up and found in the metadata before the indication is delivered, so the NoSuchObjectClass error cannot occur subsequently. If an application program attempts to return NoSuchObjectClass, CMIP services rejects the response with this error code. The other errors which fall into this category are SyncNotSupported and InvalidScope. CMIP services returns an ROER to all indications that specify a sych other than bestEffort and client agent application program s will never receive an indication that specifies atomic synchronization. Also all scopes are validated to be completely syntactically correct before the indication is delivered to the agent application. So these errors are not allowed by CMIP services. If an agent application program generates these errors, then the agent application program is in error.

Action: A program ming error exists in the client agent application. Correct the application program to send the CMIP error that actually occurred.

---

925 MB_ERR_LOST_CONNECTION

Explanation: The application program has exited.

Action: It should reconnect.

---

929 MB_ERR_LOCAL_ID_ALREADY_REGISTERED

Explanation: The local identifier is not unique.

Action: Correct the client application program to pass a unique local identifier with each MIBSendRegister API function call.

931 MB_ERR_SOURCE_NOT_IN_SUBTREE

Explanation: Only instances registered with the SUBTREE_MANAGER or EVENT_HANDLER capabilities are permitted to use the source override feature on requests and responses. For the EVENT_HANDLER application program (there may be only one), any distinguished name may be specified as the source, with no restrictions. But for subtree manager application program s, there is a restriction placed on the distinguished names which may be specified. The restriction is that the distinguished name must be within the subtree managed by the subtree manager, i.e. the distinguished name specified as the source must have the distinguished name of the subtree as a prefix. This error is returned when a subtree manager specifies a source outside its managed subtree.

Action: A programming error exists in the subtree manager application program. Correct the subtree manager application program.

932 MB_ERR_MAX_OUTSTANDING

Explanation: There are no remaining, unused invoke identifiers on this connection.

Action: Increase the value of the max outstanding invoke identifiers parameter passed to the MIBConnect function.

933 MB_ERR_CMIP_ERR_NOT_STM

Explanation: A client agent application program returned a CMIP error as its response to an indication. In checking CMIP error responses, CMIP services takes several pieces of information into account. One of them is the operation-type for the indication (e.g. m-Get, m-Action, etc). Another is the client agent's capabilities, specifically whether the instance responding had the SUBTREE_MANAGER capability set when it was registered. This error is returned to the agent by CMIP services because CMIP services only allows the given CMIP error to be returned by subtree manager agents. This is because the given error is checked for by CMIP services before the indication is delivered to the client agent application. Except in the case of subtree managers, CMIP services has already verified that the error did not occur. Since the agent is not a subtree manager, the agent should not be allowed to return this error, since it could not have occurred. Had the agent been a subtree manager the error response would have been allowed, because in that case CMIP services could not have verified ahead of time that the error did not occur. It should not be inferred from this discussion that the solution is to make the client agent a subtree manager. Instead, it should be assumed that the client incorrectly detected the error condition signaled by the response, and that the client's response is in error.

Action: Correct the client agent application program to return a valid CMIP response or an allowed CMIP error response.

934 MB_ERR_NOT_SUBTREE_MGR

Explanation: In a CMIP message, the information which determines the source of the message comes from three places. The CMIP string may specify the distinguished name of the instance sending the request or response. Or, the string may include the "&DN". macro. This is replaced by CMIP services with the distinguished name of the instance sending the message as identified by the local identifier supplied on the API call. Finally, a subtree manager instance is allowed to use the "&DN" macro and specify a source-override. This results in the distinguished name supplied on the override being substituted for the &DN rather than the DN of the subtree manager. This error indicates that the client application program specified a non-null value for the source parameter on a request or response (MIBSendResponse, MIBSendCmipRequest, MIBCmipRequest, MIBSendCmipResponse, or MIBCmipResponse) but the managed object instance sending the request or response is not a subtree manager.

Action: Correct the client application. If the application program has been designed and coded to fulfill all the responsibilities of a subtree manager, then enable the SUBTREE MANAGER on the instance's MIBRegister call. Otherwise, do not specify the source parameter on the API function call.

935 MB_ERR_DIDNT_USE_AMPER_IID

Explanation: An incorrect invoke identifier was used in a CMIP request.

Action: Use the &IID MIB variable to include the new invoke identifier for this request.

936 MB_ERR_CMIP_ERR_NOTASROIV

Explanation: An agent application program attempted to return a response containing a CMIP error. The response is allowed given the operation value of the request, but CMIP services only supports sending the CMIP error as an ROER, not as an m-Linked-Reply. This error indicates that in order to deliver the message, the response would have had to be reformatted as an m-Linked-Reply. Since CMIP services supports sending all errors as linked replies that can legitimately be returned as linked replies, this error indicates that the error should not have been sent as a not-last response.

Action: Correct the client agent application program to return the error as a final response.

937 MB_ERR_INVALID_MSG_FORMAT

Explanation: This error is returned when CMIP services cannot parse the module and type or the top-level sequence of a request or response. Incorrect values in the invoke identifier, operation-value, argument or argument-type labels can cause this error. Incorrect values in the module and type strings also cause this error.

Action: Correct the string.

938 MB_ERR_EMPTY_ROIV_INVALID

Explanation: CMIP services was processing a client agent response and needed to reformat the response into an ROIVapdu for transmission as an m-LinkedReply. But the client agent application program did not provide an argument on the response, so the response cannot be formatted as an ROIV-m-LinkedReply. Because of the possibility of this failure, and the fact that agent application programs cannot predict whether the reformatting will be required, it is required that an argument be provided on all responses. This error is only checked when the argument is actually required, so it may appear to be an intermittent problem to the client agent, nevertheless, it is actually a consistent problem.

Action: Determine which API function call was used to send the response and correct the client agent application program to provide an argument on the response in all cases.

939 MB_ERR_INVALID_RESP

Explanation: There are several checks which CMIP services makes to validate an object instance's response. The object instance supplies an invokeId, a destination (in the form of an association handle), and a response string. This error can indicate a number of different failures which all have in common that the response was invalid because that instance was not allowed to respond to the specified request at the current time. The possible failures are:

- The invoke identifier and association handle did not specify a valid indication (that is, no instance is allowed to respond to the "request", because it does not exist).
- The invoke identifier and association handle specify a valid indication, but the object instance responding is not allowed to respond to that indication because it was not a recipient of the indication.
- The invoke identifier and association handle specify a valid indication, but the object instance responding is not allowed to respond to that indication because it has already responded to that indication with a "final" response.
- The invoke identifier and associate handle specify a valid delete indication to which the object instance is allowed to respond, but the instance responded "out of phase", either sending a phase-1 response during phase 2 or vice versa.
- When CMIP services discovers that a manager application program has terminated, CMIP services removes all indications from that manager from its log. Otherwise valid agent responses to the indication are rejected with this return code. This is the only case where the client agent application program is not at fault. One possible cause for this error (in cases 2 and 3) is that the client agent application program specified the wrong local identifier on the response.

Action: Determine if the manager application program which issued the request terminated before the agent responded (see case 5 above). If so, then this error may be ignored. Otherwise correct the client agent application program to respond correctly.

941 MB_ERR_CANCELGET_RESP_INVALID

Explanation: CMIP services application program makes supporting the cancel get operation trivial for agents to implement by treating it as an unconfirmed request from the agent's viewpoint. Whenever CMIP services receives an m-CancelGet it takes care of cancelling the get and responding to the m-CancelGet and sending the operationCancelled ROER. Agents may either continue to respond to the get as if the cancel get had not been received (and CMIP services discards these responses) or agents may abort their get and send an operationCancelled error. Because of this design, agents are not allowed to respond to the m-CancelGet indication, and they receive this error if they do.

Action: Correct the client agent application program to not respond to m-CancelGet indications.

945 MB_ERR_CONNECT

Explanation: The MIBConnect was not successful. If the error condition indicated by the OPEN ACB error value parameter can be eliminated, another MIBConnect can be issued.

952 HDR_SYNTAX_ERROR

Explanation: The module and type information that must accompany all messages is wrong. The value provided either does not contain both a module and type name, separated by a period.

Action: Fix the type reference to be complete. A valid example is: `CMIP-1.ROIVapdu`.

953 INVALID_HDR_DEST_TYPE

Explanation: The type of the destination (the value for dest-type) contained in the string header is invalid. The only allowed types are:
- 0: none provided
- 1: association handle
- 2: Distinguished name of an instance
- 3: AE-title of a peer system

Action: Fix the dest-type in the string header.

954 INVALID_HDR_SRC_TYPE

Explanation: The type of the source (the value for src-type) contained in the string header is invalid. The only allowed types are:

- 0: none provided
- 1: association handle
- 2: Distinguished name of the sending instance

Normally you should be doing one of two things. If this message is a response, the src-type MUST BE 1 - association handle. If this message is a request, the type should normally be 0. The only time any other value is used is when the request is coming from a specific instance and you need to provide its name for us to resolve an &DN MIB variable.

Action: Fix the src-type in the string header.

955 UNRECOGNIZED_HDR_LABEL

Explanation: A label was encountered in the string header that was unrecognized or out of sequence.

Action: If the message was sent using MIBSendRequest or MIBSendResponse, fix the string to align with the definition of the string header contained in ISTASN1. If the message was sent by CMIP services, or using MIBSendCmipRequest or MIBSendCmipResponse, call IBM Service.

956 KEY_IS_NULL

Explanation: While parsing the string an & was encountered. The valid MIB variables are &IID, &OC, &DN, and &GTM.

Action: Fix the value to reference a valid MIB variable or avoid the use of a MIB variable or surround the value in quotation marks.

957 KEY_NOT_FOUND

Explanation: While parsing the string a MIB variable (a value which begins with the character '&') was encountered for which does not exist. The allowed values are &IID, &OC, &DN, and &GTM.

Action: Fix the value to refer to a valid MIB variable or avoid the use of a MIB variable or surround the value in quotation marks.

958 MIB_VAR_NOT_LOADED

Explanation: An invalid MIB variable was encountered.

Action: Correct the use of the MIB variable to be one of those defined, or use a real value. If you want the value to be a string that begins with &, you must surround the value in quotes.

961 LABV_END_QUOTE_NOT_FOUND

Explanation: A string was found that began with a quote (single or double) for which there was no closing quote.

Action: Fix the string value to conform to the rules for construction of string values.

962 LABV_NULL_VALUE

Explanation: CMIP services was provided with an input string that did not include a value. This is a warning that there was not a value in the string being processed.

Action: This may be working correctly, assuming the input string intended did not include a value. This is unlikely since normally it is only necessary to parse strings that contain values. Change the string to be a valid value for an ASN.1 syntax.

963 LABV_INVALID_CHAR_IN_VALUE

Explanation: An invalid character was found in a value in the string being parsed.

Action: Fix the string value to conform to the rules for construction of string values.

964 LABV_INVALID_GROUP_DELIMITER

Explanation: The only characters that are allowed to follow a right parenthesis in a string are comma and right parenthesis. Something else was encountered.

Action: Fix the string value to conform to the rules for construction of string values.

965 LABV_EMPTY_STRING

Explanation: CMIP services was handed a string with no contents. It did not return any labels or values.

Action: None; you have reached the end of the string. Processing for the string should now terminate. This is working as designed.

1000 MB_WARN_DATA_SPACE_FULL

Explanation: If using a data space and the data space is out of storage, this warning is returned to remind the application program that no messages will be returned to this application program. This message will still be routed to CMIP services.

Action: Remove messages from the data space.

1001 MB_WARN_EXIT_FAILURE

Explanation: If using common storage area storage and the application program has indicated that it has had an unrecoverable error when returning to the read queue exit routine, this warning is returned to remind the application program that no messages will be returned to the application program. This message will still be routed to CMIP services.

Action: The application program should disconnect and connect again.

1002 MB_DATA_ON_DATA_SPACE

Explanation: CMIP services has placed one or more messages in the data space.

Action: Remove messages from the data space.

1004 MB_ERR_INVALID_ARGUMENT

Explanation: The argument parameter was not provided.

Action: Correct the argument.

1005 MB_ERR_INVALID_ARGUMENT_TYPE

Explanation: An incorrect argument type parameter was provided.

Action: Correct the argument type.

1006 MB_ERR_INVALID_ASSOC_HANDLE

Explanation: An incorrect association handle parameter was provided.

Action: Correct the association handle.

1007 MB_ERR_INVALID_SMAE_NAME

Explanation: The value specified for the SMAE name buffer parameter is not valid.

Action: Correct the SMAE name.

1008 MB_ERR_CMIP_SERVICES_INACTIVE

Explanation: CMIP services is inactive.

If using common storage area storage, the read queue exit routine stops functioning.

If using data space storage, messages are not put on the data space.

Action: Start CMIP services.

1009 MB_ERR_INVALID_DS_VECTOR

Explanation: The value specified for the data space vector length parameter is valid, but the data space vector parameter is not provided.

Action: Correct the parameters.

1010 MB_ERR_INVALID_DEST_TYPE

Explanation: An incorrect destination type parameter was passed.

Action: Correct the parameter.

1011 MB_ERR_INVALID_DIST_NAME

Explanation: An incorrect distinguished name was provided.

Action: Correct the parameter.

1012 MB_ERR_INVALID_MAX_INVOKE_IDS

Explanation: The value specified for the maximum outstanding requests parameter is not valid.

Action: Correct the parameter.

1013 MB_ERR_INVALID_API_LEVEL

Explanation: An incorrect value for the API level parameter was passed.

Action: Correct the parameter.

1014 MB_ERR_INVALID_APPL_NAME

Explanation: The value specified for the application name parameter is longer than 8 characters.

Action: Correct the parameter.

1015 MB_ERR_INVALID_DS_VECTOR_SIZE

Explanation: If the data space vector parameter is specified, the data space vector length must be at least the size of (ISTRIV10_t), which is the length of the data space vector.

Action: Correct the parameter.

1016 MB_ERR_INVALID_SMAE_NAME_SIZE

Explanation: The buffer sent to the MIBConnect function is too small to accommodate the name of the SMAE. The actual amount of storage required is returned in the SMAE name length parameter.

Action: Correct the parameter.

1017 MB_ERR_INVALID_INVOKE_ID

Explanation: The invoke identifier parameter was not provided.

Action: Correct the parameter.

1018 MB_ERR_MIBDISCONNECT

Explanation: The MIBDisconnect function was not successful.

Action: If the error condition indicated by the CLOSE ACB error value parameter can be eliminated, another MIBDisconnect can be issued.

1019 MB_ERR_INVALID_MSG

Explanation: An incorrect message parameter was provided.

Action: Correct the parameter.

1020 MB_ERR_INVALID_OBJECT_CLASS

Explanation: An incorrect object class parameter was provided.

Action: Correct the parameter.

1021 MB_ERR_INVALID_READ_QUEUE_EXIT

Explanation: The read queue exit routine was not provided.

Action: Correct the parameter.

1022 MB_ERR_INVALID_SYSTEM_NAME_SIZE

Explanation: The buffer sent to the MIBConnect function is too small to accommodate the name of the system object. The actual amount of storage required is returned in the system object name buffer size parameter.

Action: Increase the buffer size.

1023 MB_ERR_INVALID_LOCAL_ID_SIZE

Explanation: The value specified on the local identifier length parameter is outside the acceptable range of 1—8.

Action: Increase the buffer size.

1024 MB_ERR_TRANSMIT

Explanation: An apparent error occurred. Either there is a logic error in VTAM, or the MIBDisconnect function has been issued, but it has not completed.

Action: Do not use any other services once MIBDisconnect has been issued.

1025 MB_ERR_VTAM_INACTIVE

Explanation: VTAM is inactive.

Action: Start VTAM.

1026 MB_ERR_INVALID_USER_DATA

Explanation: The user data parameter was not provided.

Action: Increase the buffer size.

1027 MB_ERR_INVALID_ERROR_FLAG

Explanation: The CLOSE ACB error value parameter does not point to a valid storage location.

Action: Correct the parameter.

1028 MB_ERR_INVALID_RELEASE_LEVEL

Explanation: The value specified for the VTAM release level parameter is not valid.

Action: Correct the parameter.

1029 MB_ERR_INVALID_PASSWORD

Explanation: The value specified for the password parameter is not between 0 and 8 characters.

Action: Correct the parameter.

1030 MB_ERR_INVALID_CAPABILITY_FLAGS

Explanation: The value specified for the capability flags parameter is not valid.

Action: Correct the parameter.

1031 MB_ERR_INVALID_TPEND_EXIT

Explanation: The TPEND exit routine is not valid.

Action: Correct the parameter.

1032 MB_ERR_INVALID_LAST_IN_CHAIN_FLAG

Explanation: An incorrect last in chain parameter was provided.

Action: Correct the parameter.

1033 MB_ERR_INVALID_SUCCESS_FLAG

Explanation: An incorrect success parameter was provided.

Action: Correct the parameter.

1034 MB_ERR_INVALID_SYSTEM_NAME

Explanation: The value specified for the system object name buffer parameter is not valid.

Action: Correct the parameter.

1035 MB_ERR_INVALID_CONNECT_OPTIONS

Explanation: The value specified on the connection options parameter is not valid. Specify either NO_CONNECT_OPTIONS or SHORT_NAMES as the value for the connection options parameter.

Action: Correct the parameter.

1036 MB_ERR_INVALID_NAME_TYPE

Explanation: An incorrect name type parameter was provided.

Action: Correct the parameter.

1037 MB_ERR_INVALID_NAME_BINDING

Explanation: An incorrect name binding parameter was provided.

Action: Correct the parameter.

1038 MB_ERR_INVALID_ALLOMORPHS_COUNT

Explanation: An incorrect allomorphs count parameter was provided.

Action: Correct the parameter.

1039 MB_ERR_INVALID_ALLOMORPHS_ARRAY

Explanation: An incorrect allomorphs array parameter was provided.

Action: Correct the parameter.

1040 MB_ERR_INVALID_CREATE_HANDLERS_COUNT

Explanation: An incorrect create handlers count parameter was provided.

Action: Correct the parameter.

1041 MB_ERR_INVALID_CREATE_HANDLERS_ARRAY

Explanation: An incorrect create handlers array parameter was provided.

Action: Correct the parameter.

1042 MB_ERR_INVALID_LOCAL_ID

Explanation: An incorrect local identifier parameter was provided.

Action: Correct the parameter.

1043 MB_ERR_INVALID_DEST

Explanation: The value of the destination parameter is inconsistent with the value of the destination type parameter. This return code is returned if, for example, destination type is DS_ASSOC_HANDLE, but destination is NULL.

Action: Correct the parameter.

CMER VIT entry error codes

These error codes can appear only in CMER VIT entries.

151

Explanation: An invalid parameter was received.

Action: No action is required. Other errors logged in CMER VIT entries or sent to an application program may indicate the cause of the problem.

153

Explanation: An error was encountered by notification services.

Action: No action is required. Other errors logged in CMER VIT entries or sent to an application program may indicate the cause of the problem.

156

Explanation: A CMIP services dataset could not be opened.

Action: Check the VTAM JCL to ensure that required DD cards are present and point to the correct datasets. Check the datasets to verify the presence of the required members. Then restart CMIP Services using the MODIFY VTAMOPTS,OSIMGMT=YES command.

157

Explanation: A CMIP services dataset contains incorrect data.

Action: Reload the CMIP services datasets to ensure that the datasets are not corrupted. Then restart CMIP Services using the MODIFY VTAMOPTS,OSIMGMT=YES command.

158

Explanation: The directory definition file contained a syntax error.

Action: Correct the directory definition file and restart CMIP services.

159

Explanation: The name attribute in the directory definition file was invalid.

Action: Correct the directory definition file and restart CMIP services.

161

Explanation: The name attribute in the directory definition file was missing.

Action: Correct the directory definition file and restart CMIP services.

162

Explanation: An attribute in the directory definition file was listed more than once in the same entry.

Action: Correct the directory definition file and restart CMIP services.

166

Explanation: The class attribute in the directory definition file was invalid.

Action: Correct the directory definition file and restart CMIP services.

167

Explanation: A generic error occurred. Other error codes should be traced or returned to the user in a MIB.ServiceError message.

Action: No action is required. Other errors logged in CMER VIT entries or sent to an application program may indicate the cause of the problem.

168

Explanation: The class attribute in the directory definition file was missing.

Action: Correct the directory definition file and restart CMIP services.

174

Explanation: A CMIP services dataset contains incorrect data.

Action: Reload the CMIP services datasets to ensure that the datasets are not corrupted. Then restart CMIP Services using the MODIFY VTAMOPTS,OSIMGMT=YES command.

1051

Explanation: An EFD filter contained too many object classes to be recognized for topology agent processing. VTAM topology agent will not generate notifications for this EFD if the OSIEVENT start option is set to PATTERNS.

Action: If the OSIEVENT start option is set to PATTERNS and the EFD which led to this warning is meant to collect information from VTAM topology agent, then the filter in the EFD must be rewritten to refer only to objects of a single class.

1052

Explanation: An EFD filter contained too many distinguished names to be recognized for topology agent processing. VTAM topology agent will not generate notifications for this EFD if the OSIEVENT start option is set to PATTERNS.

Action: If the OSIEVENT start option is set to PATTERNS and the EFD which led to this warning is meant to collect information from VTAM topology agent, then the filter in the EFD must be rewritten to refer only to a single DN.

1053

Explanation: An EFD filter contained a resource name which was too long. VTAM topology agent will not generate notifications for this EFD if the OSIEVENT start option is set to patterns.

Action: If the OSIEVENT start option is set to PATTERNS and the EFD which led to this warning is meant to collect information from VTAM topology agent, then the filter in the EFD must be rewritten to correct the object names.

1054

Explanation: An EFD destination was incorrect. The EFD will not be created.

Action: The specified destination attribute is invalid and must be changed. The destination of an EFD should be an AE registered by the application program.

1055

Explanation: An EFD filter contained an object class which was not recognized for topology agent processing. VTAM topology agent will not generate notifications for this EFD if the OSIEVENT start option is set to PATTERNS.

Action: If the OSIEVENT start option is set to PATTERNS and the EFD which led to this warning is meant to collect information from VTAM topology agent, then the filter in the EFD must be rewritten to specify a support object class.

1056

Explanation: An EFD filter was not recognized for topology agent processing. VTAM topology agent will not generate notifications for this EFD if the OSIEVENT start option is set to PATTERNS.

Action: If the OSIEVENT start option is set to PATTERNS and the EFD which led to this warning is meant to collect information from VTAM topology agent, then the filter in the EFD must be rewritten to follow a recognizable pattern.

1057

Explanation: An EFD filter was recognized as having nothing to do with VTAM topology. VTAM topology agent will not generate notifications for this EFD regardless of the setting of the OSIEVENT start option.

Action: If the EFD which led to this warning is meant to collect information from VTAM topology agent, then the filter in the EFD must be rewritten, as it seems to have nothing to do with VTAM topology.

Appendix D. VTAM CMIP services compliance with related standards and profiles

This section is designed to help you understand how VTAM CMIP services complies to the standards related to OSI systems management.

VTAM CMIP services implements functions that are defined in International Standards Organization (ISO) standards documents and industry profiles.

ISO standards documents

This section indicates how VTAM CMIP services conforms to several ISO standards related to OSI systems management.

ISO 9596-1 CMIP—Common Management Information Protocol

VTAM CMIP services implements all functional units specified for CMIP Version 2 in this standard. Atomic synchronization is not supported.

(ISO 10164-5) OSI systems management part 5: event report function

VTAM CMIP services implements the event forwarding discriminator (EFD) described in this standard. All of the object management functions specified for the EFD are supported (GET, SET, CREATE, DELETE). VTAM CMIP services supports general discriminator constructs of any complexity. VTAM CMIP services does not support any of the conditional packages defined for the class or substring operations on SET valued attributes.

ISO 8650 ACSE—Association Control Service Element

VTAM CMIP services implements all required aspects of the protocol specified as ACSE Version 1 in this standard. VTAM CMIP services accepts all elements of protocol specified, but only a specific set of parameters are actually used.

ISO 8823 presentation layer

VTAM CMIP services implements all required aspects of the presentation layer protocol used for establishing and releasing connections. VTAM CMIP services also implements the encoding and decoding function specified. It supports a single transfer syntax, basic encoding rules (BER). Any other transfer syntaxes are rejected. If the partner does not support BER for an abstract syntax, an association cannot be established.

ISO 8825 BER—Basic Encoding Rules (BER)

VTAM CMIP services supports encoding and decoding of all of the ASN.1 types using the basic encoding rules. Some of the types are supported to a limited extent, specifically:

- Integers are encoded and decoded only up to the size supported by the machine in a native format. When any larger integers are received, they are left in the BER form, and passed to the user in the BER form.
- Only the default code page is supported for GraphicString.

ISO standards documents

This section indicates how VTAM CMIP services conforms to several industry profiles governing the implementation of ISO standards. These profiles are defined to allow interoperation between different implementations of the standards. Each covers a specific set of standards and specifies the set of mandatory and optional elements of those standards. Each profile specifies value ranges, message sizes, and so on, that ensure a common implementation base.

DISP 11183-1, AOM 10

This profile governs the implementation of the ACSE, presentation layer, session layer for use with Remote Operation Service Element (ROSE) and Common Management Interface Service Element (CMISE).

VTAM CMIP services implements the relevant portions of this profile. All required elements of protocol are supported.

DISP 11183-3, AOM 12

This profile governs the implementation of the CMISE.

VTAM CMIP services implements the relevant portions of this profile.

AOM221—general event report management

This profile governs the implementation of the event forwarding discriminator object class, which VTAM CMIP services supports.

This profile specifies a minimum set of attributes that must be permitted to appear in discriminator constructs and the minimum levels of complexity that must be supported.

VTAM CMIP services complies; VTAM CMIP services allows any level of complexity and supports any set of events (GDMO NOTIFICATION templates and the associated attribute templates) with which it is loaded.

The profile also requires support for all matching rules that can be specified in the discriminator construct. VTAM CMIP services does not support the SET operations: subset, superset, and non-null-set-intersection.

The profile also requires support for two non-mandatory packages: weekly scheduling and backup destinations. VTAM CMIP services supports neither.

This profile does not require support for confirmed mode conditional, which VTAM CMIP services does not support.

Appendix E. VTAM topology agent object and attribute tables

VTAM-supported objects for snapshot operations

The set of objects VTAM supports for *snapshot* operations is presented in the following table.

Table 22. Supported object classes for snapshot

Object identifier	Object name
1.3.18.0.0.1811	*luCollection*
1.3.18.0.0.2291	*logicalUnitIndex*
1.3.18.0.0.2152	*snaLocalTopo*
1.3.18.0.0.2151	*snaNetwork*

Naming attributes for snapshot objects

Naming attributes for snapshot objects are presented in the following table.

Table 23. Naming attributes for snapshot objects

Attribute identifier	Attribute name	Object name
1.3.18.0.0.2216	*graphId*	*snaLocalTopo*
1.3.18.0.0.2216	*graphId*	*snaNetwork*
1.3.18.0.0.1815	*luCollectionId*	*luCollection*
1.3.18.0.0.1815	*logicalUnitIndexName*	*logicalUnitIndex*

VTAM-supported objects for snapshot responses

The set of objects VTAM supports for the *snapshot* operation responses includes all valid objects for a GET or *snapshot* request and the objects in the following table.

Table 24. Unique objects for snapshot response

Object identifier	Object name	Snapshot type
1.3.18.0.0.2278	*crossDomainResourceManager*	*snaNetwork*
1.3.18.0.0.1848	*virtualRoute*	*snaNetwork*
1.3.18.0.0.1849	*virtualRoutingNode*	*snaNetwork, snaLocalTopo*
1.3.18.0.0.1840	*subareaTransmissionGroup*	*snaLocalTopo*
1.3.18.0.0.1823	*appnTransmissionGroup*	*snaNetwork, snaLocalTopo*

VTAM-supported attributes for snapshot responses

The set of attributes VTAM supports for the *snapshot* operation responses includes all valid attributes for GET operations and the attributes in the following table.

Table 25. Unique attributes for snapshot response

Attribute identifier	Attribute name	Snapshot type
1.3.18.0.0.5246	*realSSCPname*	*snaNetwork*
1.3.18.0.0.1958	*cp-cpSessionSupport*	*snaNetwork, snaLocalTopo*
1.3.18.0.0.1941	*appnTGcapabilities*	*snaNetwork, snaLocalTopo*

VTAM-supported objects for GET operation

The set of objects VTAM supports for the GET operation is presented in the following table.

Table 26. Supported object classes for GET

Object identifier	Object name
1.3.18.0.0.2281	*crossDomainResource*
1.3.18.0.0.2267	*definitionGroup*
1.3.18.0.0.1821	*appnEN*
1.3.18.0.0.1826	*interchangeNode*
1.3.18.0.0.1827	*lenNode*
1.3.18.0.0.2085	*logicalLink*
1.3.18.0.0.1829	*logicalUnit*
1.3.18.0.0.1803	*luGroup*
1.3.18.0.0.1833	*migrationDataHost*
1.3.18.0.0.1822	*appnNN*
1.3.18.0.0.2089	*port*
1.3.18.0.0.2288	*appnRegisteredLu*
1.3.18.0.0.1843	*t2-1Node*
1.3.18.0.0.1844	*t4Node*
1.3.18.0.0.1845	*t5Node*

VTAM-supported attributes for GET operation

The set of mandatory attributes supported for the GET operation for a given object is presented in the following tables. There is one table per supported object class for the GET operation.

Table 27. CDRSC attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
2.9.3.2.7.33	*availabilityStatus*
1.3.18.0.0.3591	*cdrscRealLUname*

Table 27. CDRSC attribute table (continued)

Attribute identifier	Attribute name
1.3.18.0.0.2194	*dependencies*
1.2.124.360501.1.240	*functionID*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
1.3.18.0.0.2284	*nlrResidentNodePointer* (naming attribute)
1.3.18.0.0.2276	*nonLocalResourceName*
1.3.18.0.0.2277	*nonLocalResourceType*
2.9.3.2.7.65	*objectClass*
2.9.3.2.7.35	*operationalState*
1.3.14.2.2.4.35	*opNetworkName*
2.9.3.2.7.66	*packages*
2.9.3.2.7.36	*proceduralStatus*
1.2.124.360501.1.302	*supportedResources*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*
0.0.13.3100.0.7.50	*userLabel*

Table 28. Definition group attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.33	*availabilityStatus*
2.9.3.2.7.50	*allomorphs*
1.3.18.0.0.2272	*definitionGroupName* (naming attribute)
1.3.18.0.0.2194	*dependencies*
1.2.124.360501.1.240	*functionID*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
2.9.3.2.7.65	*objectClass*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
1.2.124.360501.1.302	*supportedResources*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Table 29. APPN end node attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
1.2.124.360501.1.209	*attachedCircuitList*
2.9.3.2.7.33	*availabilityStatus*

Table 29. APPN end node attribute table (continued)

Attribute identifier	Attribute name
1.3.18.0.0.2194	*dependencies*
1.2.124.360501.1.240	*functionID*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
1.3.18.0.0.1997	*nnServerPointer*
2.9.3.2.7.65	*objectClass*
1.3.14.2.2.4.33	*opEquipmentList*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
2.9.3.2.7.36	*proceduralStatus*
1.3.18.0.0.2032	*snaNodeName* (naming attribute)
1.3.14.2.2.4.53	*softwareList*
1.2.124.360501.1.302	*supportedResources*
1.3.18.0.0.2296	*sysplexInfo*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Table 30. Interchange node attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
1.3.18.0.0.1940	*appnNodeCapabilities*
1.2.124.360501.1.209	*attachedCircuitList*
2.9.3.2.7.33	*availabilityStatus*
1.3.18.0.0.2194	*dependencies*
1.3.18.0.0.2025	*dlurList*
1.3.18.0.0.1967	*erList*
1.3.18.0.0.1970	*extendedAppnNodeCapabilities*
1.2.124.360501.1.240	*functionID*
1.3.18.0.0.1972	*gatewaySSCP*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
2.9.3.2.7.65	*objectClass*
1.3.14.2.2.4.33	*opEquipmentList*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
2.9.3.2.7.36	*proceduralStatus*
1.3.18.0.0.2013	*puName*
1.3.18.0.0.2019	*resourceSequenceNumber*
1.3.18.0.0.2020	*routeAdditionResistance*

Table 30. Interchange node attribute table (continued)

Attribute identifier	Attribute name
1.3.18.0.0.2032	*snaNodeName* (naming attribute)
1.3.14.2.2.4.53	*softwareList*
1.3.18.0.0.2035	*subareaAddress*
1.3.18.0.0.2036	*subareaLimit*
1.2.124.360501.1.302	*supportedResources*
1.3.18.0.0.2296	*sysplexInfo*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Table 31. Low-entry networking node attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
1.2.124.360501.1.209	*attachedCircuitList*
2.9.3.2.7.33	*availabilityStatus*
1.3.18.0.0.2194	*dependencies*
1.2.124.360501.1.240	*functionID*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
2.9.3.2.7.65	*objectClass*
1.3.14.2.2.4.33	*opEquipmentList*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
2.9.3.2.7.36	*proceduralStatus*
1.3.18.0.0.2032	*snaNodeName* (naming attribute)
1.3.14.2.2.4.53	*softwareList*
1.2.124.360501.1.302	*supportedResources*
1.3.18.0.0.2296	*sysplexInfo*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Table 32. Logical link attribute table

Attribute identifier	Attribute name
1.3.18.0.0.2119	*adjacentLinkStationAddress*
1.3.18.0.0.2122	*adjacentNodeName*
1.3.18.0.0.2121	*adjacentNodeType*
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
2.9.3.2.7.33	*availabilityStatus*
2.9.3.5.7.1	*connectionID*

Table 32. Logical link attribute table (continued)

Attribute identifier	Attribute name
1.3.18.0.0.2125	*connectionType*
1.3.18.0.0.2194	*dependencies*
1.3.18.0.0.7899	*dlurLocalLsAddress*
1.3.18.0.0.2309	*dlurName*
1.3.18.0.0.2235	*endpointForArc*
1.2.124.360501.1.240	*functionID*
1.3.18.0.0.2131	*lineType*
1.3.18.0.0.2133	*linkName* (naming attribute)
1.3.18.0.0.2134	*linkStationRole*
1.3.18.0.0.2137	*maxBTUsize*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
2.9.3.2.7.65	*objectClass*
1.3.14.2.2.4.33	*opEquipmentList*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
1.3.18.0.0.2236	*partnerConnection*
1.3.18.0.0.2142	*portId*
2.9.3.2.7.36	*proceduralStatus*
1.2.124.360501.1.302	*supportedResources*
1.3.18.0.0.2045	*transmissionGroupNumber*
2.9.3.5.7.14	*underlyingConnectionNames*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Table 33. Logical unit attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
1.2.124.360501.1.209	*attachedCircuitList*
2.9.3.2.7.33	*availabilityStatus*
1.3.18.0.0.2194	*dependencies*
1.2.124.360501.1.240	*functionID*
1.3.18.0.0.1984	*luName* (naming attribute)
1.3.18.0.0.1819	*luSecondName*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
2.9.3.2.7.65	*objectClass*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*

Table 33. Logical unit attribute table (continued)

Attribute identifier	Attribute name
2.9.3.2.7.36	*proceduralStatus*
1.3.18.0.0.2018	*residentNodePointer*
1.2.124.360501.1.302	*supportedResources*
1.3.18.0.0.7900	*tn3270ClientDnsName*
1.3.18.0.0.7901	*tn3270ClientIpAddress*
1.3.18.0.0.7902	*tn3270ClientportNumber*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*
0.0.13.3100.0.7.50	*userLabel*

Table 34. LU group attribute table

Attribute identifier	Attribute name
2.9.3.2.7.50	*allomorphs*
1.3.18.0.0.1808	*luGroupMembers*
1.3.18.0.0.1807	*luGroupName* (naming attribute)
1.3.18.0.0.1809	*luGroupSize*
2.9.3.2.7.63	*nameBinding*
2.9.3.2.7.65	*objectClass*
2.9.3.2.7.66	*packages*

Table 35. Migration data host node attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
1.2.124.360501.1.209	*attachedCircuitList*
2.9.3.2.7.33	*availabilityStatus*
1.3.18.0.0.2194	*dependencies*
1.3.18.0.0.1967	*erList*
1.2.124.360501.1.240	*functionID*
1.3.18.0.0.1972	*gatewaySSCP*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
1.3.18.0.0.1997	*nnServerPointer*
2.9.3.2.7.65	*objectClass*
1.3.14.2.2.4.33	*opEquipmentList*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
2.9.3.2.7.36	*proceduralStatus*
1.3.18.0.0.2013	*puName*
1.3.18.0.0.2032	*snaNodeName* (naming attribute)

Table 35. Migration data host node attribute table (continued)

Attribute identifier	Attribute name
1.3.14.2.2.4.53	*softwareList*
1.3.18.0.0.2035	*subareaAddress*
1.3.18.0.0.2036	*subareaLimit*
1.2.124.360501.1.302	*supportedResources*
1.3.18.0.0.2296	*sysplexInfo*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Table 36. APPN network node attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
1.3.18.0.0.1940	*appnNodeCapabilities*
1.2.124.360501.1.209	*attachedCircuitList*
2.9.3.2.7.33	*availabilityStatus*
1.3.18.0.0.2194	*dependencies*
1.3.18.0.0.2025	*dlurList*
1.3.18.0.0.1970	*extendedAppnNodeCapabilities*
1.2.124.360501.1.240	*functionID*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
2.9.3.2.7.65	*objectClass*
1.3.14.2.2.4.33	*opEquipmentList*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
2.9.3.2.7.36	*proceduralStatus*
1.3.18.0.0.2019	*resourceSequenceNumber*
1.3.18.0.0.2020	*routeAdditionResistance*
1.3.18.0.0.2032	*snaNodeName* (naming attribute)
1.3.14.2.2.4.53	*softwareList*
1.2.124.360501.1.302	*supportedResources*
1.3.18.0.0.2296	*sysplexInfo*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Table 37. Port attribute table

Attribute identifier	Attribute name
1.3.18.0.0.2115	*abmSupported*
1.3.18.0.0.2117	*adapterAddresses*
1.3.18.0.0.2118	*adapterNumbers*

Table 37. Port attribute table (continued)

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
2.9.3.2.7.33	*availabilityStatus*
2.9.3.5.7.1	*connectionID*
1.3.18.0.0.2194	*dependencies*
1.3.18.0.0.2127	*dlcName*
1.3.18.0.0.2235	*endpointForArc*
1.2.124.360501.1.240	*functionID*
1.3.18.0.0.2129	*limitedResource*
1.3.18.0.0.2130	*limitedResourceTimeout*
1.3.18.0.0.2131	*lineType*
1.3.18.0.0.2134	*linkStationRole*
1.3.18.0.0.2137	*maxBTUsize*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
2.9.3.2.7.65	*objectClass*
1.3.14.2.2.4.33	*opEquipmentList*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
1.3.18.0.0.2236	*partnerConnection*
1.3.18.0.0.2142	*portId* (naming attribute)
2.9.3.2.7.36	*proceduralStatus*
1.3.18.0.0.2146	*receiveWindowSize*
1.3.18.0.0.2244	*relatedAdapter*
1.3.18.0.0.2148	*sendWindowSize*
1.2.124.360501.1.302	*supportedResources*
2.9.3.5.7.14	*underlyingConnectionNames*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Table 38. APPN registered LU attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
2.9.3.2.7.33	*availabilityStatus*
1.3.18.0.0.2194	*dependencies*
1.2.124.360501.1.240	*functionID*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
1.3.18.0.0.2284	*nlrResidentNodePointer*

Table 38. APPN registered LU attribute table (continued)

Attribute identifier	Attribute name
1.3.18.0.0.2276	*nonLocalResourceName* (naming attribute)
1.3.18.0.0.2277	*nonLocalResourceType*
2.9.3.2.7.65	*objectClass*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
2.9.3.2.7.36	*proceduralStatus*
1.3.18.0.0.2273	*registeredBy*
1.2.124.360501.1.302	*supportedResources*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Table 39. Type 2.1 node attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
1.2.124.360501.1.209	*attachedCircuitList*
2.9.3.2.7.33	*availabilityStatus*
1.3.18.0.0.2194	*dependencies*
1.2.124.360501.1.240	*functionID*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
2.9.3.2.7.65	*objectClass*
1.3.14.2.2.4.33	*opEquipmentList*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
2.9.3.2.7.36	*proceduralStatus*
1.3.18.0.0.2032	*snaNodeName* (naming attribute)
1.3.14.2.2.4.53	*softwareList*
1.2.124.360501.1.302	*supportedResources*
1.3.18.0.0.2296	*sysplexInfo*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Table 40. Type 4 node attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
1.2.124.360501.1.209	*attachedCircuitList*
2.9.3.2.7.33	*availabilityStatus*
1.3.18.0.0.2194	*dependencies*

Table 40. Type 4 node attribute table (continued)

Attribute identifier	Attribute name
1.3.18.0.0.1967	*erList*
1.2.124.360501.1.240	*functionID*
1.3.18.0.0.1971	*gatewayNode*
1.3.18.0.0.1978	*interconnectedNetids*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
2.9.3.2.7.65	*objectClass*
1.3.14.2.2.4.33	*opEquipmentList*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
2.9.3.2.7.36	*proceduralStatus*
1.3.18.0.0.2032	*snaNodeName* (naming attribute)
1.3.18.0.0.2035	*subareaAddress*
1.3.18.0.0.2036	*subareaLimit*
1.2.124.360501.1.302	*supportedResources*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Table 41. Type 5 node attribute table

Attribute identifier	Attribute name
2.9.3.2.7.31	*administrativeState*
2.9.3.2.7.50	*allomorphs*
1.2.124.360501.1.209	*attachedCircuitList*
2.9.3.2.7.33	*availabilityStatus*
1.3.18.0.0.2194	*dependencies*
1.3.18.0.0.1967	*erList*
1.2.124.360501.1.240	*functionID*
1.3.18.0.0.1972	*gatewaySSCP*
2.9.3.2.7.63	*nameBinding*
1.3.18.0.0.2080	*nativeStatus*
2.9.3.2.7.65	*objectClass*
1.3.14.2.2.4.33	*opEquipmentList*
2.9.3.2.7.35	*operationalState*
2.9.3.2.7.66	*packages*
2.9.3.2.7.36	*proceduralStatus*
1.3.18.0.0.2013	*puName*
1.3.18.0.0.2032	*snaNodeName* (naming attribute)
1.3.14.2.2.4.53	*softwareList*
1.3.18.0.0.2035	*subareaAddress*
1.3.18.0.0.2036	*subareaLimit*

Table 41. Type 5 node attribute table (continued)

Attribute identifier	Attribute name
1.2.124.360501.1.302	*supportedResources*
1.3.18.0.0.2296	*sysplexInfo*
2.9.3.2.7.38	*unknownStatus*
2.9.3.2.7.39	*usageState*

Appendix F. VTAM topology agent attributes definition

For each attribute, the following table explains:

- ASN.1 syntax used for that attribute
- The information that attribute describes about the resource; for example, its DLC address
- What VTAM resource that attribute is referring to
- Which CMIP operations can report that attribute
- Which OSI classes that attribute applies to

abmSupported

Syntax

BOOLEAN

TRUE Supports asynchronous balance mode

FALSE

Does not support asynchronous balance mode

Meaning

Whether asynchronous balanced mode is supported

Source

XID3. This value is only TRUE when the XID format 3 received from a type 2.1 node indicates asynchronous balanced mode.

Operations

GET

Attribute of

port

adapterAddresses

Syntax

SET OF OCTET STRING

Meaning

Local DLC address; for example, local MAC/SAP address.

Source

Dependent on resource type:

NTRI physical line

The local MAC/SAP address returned as 14 characters; for example, `11223344556601`. The first 12 characters are the MAC address and the last two are the SAP address. This is the value coded on the LOCADD operand of the LINE definition statement in an NCP major node.

Note: The local MAC/SAP address does not apply to NTRI logical lines.

LAN or ATM LAN emulation switched line

The local MAC/SAP address for the XCA adapter associated with the line. This information is available only when the line and PU are active and the X'57' DLC address vector has been received.

LAN or ATM LAN emulation leased line

The local MAC/SAP address for the XCA adapter associated with

the line. This information is available only when the line and PU are active and the X'57' DLC address vector has been received.

ATM native SVC (switched line) or PVC (nonswitched line)
The local ATM address for the IBM S/390 Open Systems Adapter associated with the SVC or PVC. The local ATM address is returned as a variable length character string. The following is an example of an ATM address:

```
XXXXYYYYZZ...ZZ
```

where:

XXXX Represents the address type and plan and can be:

X'0101'
Indicates public E164 address, which means the address is in a public ATM network.

X'0002'
Indicates International Organization for Standardization (ISO) network service access point (NSAP), which means the address is in a private ATM network.

YYYY Represents the length of the address. The address can be up to 20 bytes in hexidecimal format.

ZZ...ZZ
Represents the actual ATM address. The address can be up to 20 bytes in length.

XCF line
The XCF token of the agent VTAM returned as 16 characters.

Operations
GET, SNAPSHOT(snaLocalTopo)

Attribute of
port

adapterNumbers

Syntax
SET OF INTEGER (0..65535)

Meaning
Address or addresses used to access the port

Source
Dependent on resource type:

Channel lines
This is the *decimal* representation of the channel unit address coded on the ADDRESS operand of the LINE definition statement.

Multipath channel
This is the *decimal* representation of each read and write channel unit address coded on the READ and WRITE operands of the LINE definition statement in the MPC group.

APPN host-to-host channel
This is the *decimal* representation of each read and write channel unit address coded on the READ and WRITE operands of the transport resource list entry (TRLE) associated with the PU.

NCP SDLC lines
This is the *decimal* representation of the line address coded on ADDRESS operand of the LINE definition statement.

XCA lines
The *decimal* channel unit address of the channel that connects VTAM to the 3172 Interconnect Controller.

Operations
GET, SNAPSHOT(snaLocalTopo-appnOnly)

Attribute of
port

---

adjacentLinkStationAddress

Syntax
CHOICE { IsAddr OCTET STRING, noLSaddr NULL }

Meaning
DLC address for the remote PU.

For SDLC
SDLC polling address

For token ring and frame relay
Remote MAC/SAP address

For ATM native SVCs
Destination ATM address

For ATM native PVCs
Null string

For XCF
XCF token of the adjacent VTAM

Source
Dependent on resource type:

For SDLC non-switched PUs
The SDLC polling address of the PU. This is specified on the ADDR operand of the PU statement.

For NTRI logical switched PUs
The MAC/SAP address of the remote PU in the form `11223344556601`. The first 12 characters are the MAC address and the last two are the SAP address. This information is available only when the line and PU are active and the X'57' DLC address vector has been received.

For NTRI logical subarea PUs
The MAC/SAP address of the remote link station. The MAC address is coded on the LOCADD operand of the LINE. The SAP address for NCP NTRI is X'04'.

LAN or ATM LAN emulation peripheral connections (switched)
The MAC/SAP address of the remote PU. This information is available only when the line and PU are active and the X'57' DLC address vector has been received.

LAN or ATM LAN emulation subarea connections (leased)
The MAC/SAP address of the remote link station. This MAC address is defined on the MACADDR operand of the PU definition statement. The SAP address is defined on the SAPADDR operand of the PU definition statement.

ATM native connections (TGs over SVCs)
The destination ATM address for the remote node associated with the SVC. The destination ATM address The remote ATM address is returned as a variable length character string. The following is an example of an ATM address:

```
XXXXYYYYZZ...ZZ
```

where:

XXXX Represents the address type and plan and can be:

X'0101'
Indicates public E164 address, which means the address is in a public ATM network.

X'0002'
Indicates International Organization for Standardization (ISO) network service access point (NSAP), which means the address is in a private ATM network.

YYYY Represents the length of the address. The address can be up to 20 bytes in hexidecimal format.

ZZ...ZZ
Represents the actual ATM address. The address can be up to 20 bytes in length.

ATM native connections (TGs over PVCs)
The destination ATM address for the remote node associated with the PVC is unknown.

XCF connections
The XCF token of the adjacent VTAM returned as 16 characters. This information is available only when the XCF connection is active.

Operations
GET, SNAPSHOT (snaLocalTopo)

Attribute of
logicalLink

---

adjacentNodeName

Syntax
GraphicString (SIZE(0..17))

Meaning
Network qualified name of node connected to this logicalLink.

Source
Generally, this information is available only when the line and PU are active. This represents the name of the adjacent node and depends on the type of connection (subarea or APPN) and the code level of the contacted node. These nodes are capable of providing the X'0EF1', X'0EF4', and X'0EF7' control vectors during CONTACT processing.

Subarea connection to VTAM V4R3 or later
The SSCP name of the VTAM contacted by this link station.

Subarea connection to NCP V7R1 or later
The PU name of the NCP contacted by this link station. This might be the same as the NCP load module name.

Subarea connection to backlevel subarea node (NCP earlier than V7R1 or VTAM earlier than V4R3)
The name is represented as a character string with the decimal subarea number of the contacted node. For example, `00000123` would be the name for a backlevel subarea node with subarea number 123.

APPN connections
The CP name of the contacted APPN node.

LEN connections
The CP name of the contacted LEN node or the predefined CP name or the name of the VTAM host CP supporting the LEN connection.

Operations
GET

Attribute of
logicalLink

adjacentNodeType

Syntax
ENUMERATED { unknown,

len,
nn,
en,
t1,
t20,
t4,
t5,
t21 }

Meaning
Type of attached PU or node.

Source
The node type is provided as it is currently known according to definitions at VTAM topology agent host. This attribute is related to the PUTYPE and XID operand of the PU definition statement. The value provided may change after the node is contacted and VTAM determines the actual node type of the contacted node.

PUTYPE	XID	AdjacentNodeType
1	N/A	t1
2	NO	t20
2	YES	t21 (if not yet contacted)
2	YES	len (contacted len node)
2	YES	en (contacted APPN end node)
2	YES	nn (contacted APPN network node)
4	N/A	t4 (contacted PU type 4)
4	N/A	t5 (contacted PU type 5)
5	N/A	t5 (contacted PU type 5)

Note: logicalLinks represent either a subarea or APPN connection to an adjacent node. Therefore, the adjacent node type is never be a

composite of subarea and APPN (for example, an interchange node). The manager application program must infer the actual node type of composite nodes or consult an agent application program at the node in question.

Operations
GET, SNAPSHOT(snaLocalTopo)

Attribute of
logicalLink

administrativeState

Syntax
Two types: ENUMERATED { locked,

unlocked,
shuttingDown }

OCTET: X'00'= locked

X'00'= locked
X'01'= unlocked
X'02'= shuttingDown
X'FF'= unchanged

ENUMERATED is used for GET operations. OCTET is used for SNAPSHOT operations.

Meaning
OSI administrative state.

Source
Value is always “unlocked”.

Operations
GET, SNAPSHOT(all types)

Attribute of
all objects

allomorphs

Syntax
SET OF ObjectClass (OIs)

Meaning
Classes for which this class can emulate. Constant value; depends on object class.

Source
Depends on the object class

Object class
allomorphs
t5Node
(lenNode, t2-1Node)
appnNN
(lenNode, t2-1Node)
appnEN
(lenNode, t2-1Node)
interchangeNode
(lenNode, t2-1Node, t5Node, appnNN)
migrationDataHost
(lenNode, t2-1Node, t5Node, appnEN)

others ()

Operations
GET

Attribute of
all objects

appnNodeCapabilities

Syntax
OCTET STRING (SIZE(2))

Meaning
SNA control vector 45, subfield 80:

Bit	Meaning
1...	Gateway function supported
.1..	Directory server function supported
..1.	Intermediate routing function supported
...1	Chain function supported
.... 00..	Reserved
.... ..00	SNA node type 5
.... ..11	SNA node type 2.1
1...	Release 1 border node
.1..	Interchange node
..1.	Release 2 border node
...0 0...	No HPR support
...0 1...	HPR base support
...1 0...	HPR base and tower support
...1 1...	Reserved
.... .000	Reserved

Source
This information is provided only for the node running the VTAM topology agent and only when the node is capable of being an APPN network node.

Operations
GET, SNAPSHOT(snaNetwork, snaLocalTopo)

Attribute of
interchangeNode
appnNN

appnTGcapabilities

Syntax

OCTET STRING (SIZE(1))

Meaning

TG capabilities of an APPN transmission group from SNA control vector 46, subfield 80, flags byte

Bit	Meaning
1...	tgPartnerIsAConnectionNetwork
.1..	Peripheral TG
..1.	tgPartnerType is type 2
..0.	tgPartnerType is type 2.1
...0 0...	tgType is boundary function or APPN TG
...0 1...	tgType is interchange TG
...1 0...	tgType is virtual route TG
...1 1...	Reserved
.... .1..	intersubnetworkLink for Release 1 border nodes
.... .0..	intersubnetworkLink for Release 2 border nodes
.... ..1.	Reserved
....1	Reserved

Source

CV 46, subfield 80 for active APPN TGs. This attribute does not apply to LEN connections.

Operations

SNAPSHOT(snaNetwork,snaLocalTopo)

Attribute of

appnTransmissionGroup

attachedCircuitList

Syntax

SET OF ObjectInstance

Meaning

VTAM always builds the empty set, ().

Source

Not supported

Operations

GET

Attribute of

appnEN
interchangeNode
lenNode
logicalUnit
migrationDataHost
appnNN

t2-1Node
t4Node
t5Node

availabilityStatus

Syntax

SET OF INTEGER { inTest (0),

failed (1),
powerOff (2),
offLine (3),
offDuty (4),
dependency (5),
degraded (6),
notInstalled (7),
logFull (8) }

OCTET: X'00'= no Status

X'01'= notInstalled
X'02'= degraded
X'04'= dependency
X'08'= offDuty
X'10'= offLine
X'20'= powerOff
X'40'= failed
X'80'= inTest
X'FF'= no change

INTEGER is used for GET and NOTIFICATION operations.

OCTET is used for SNAPSHOT operations.

Meaning

OSI availability status: The following values can be returned by VTAM: offline, failed, intest, dependency, degraded, and no information NULL.

Source

Determined from VTAM resource definition table entry (RDTE) finite state machine (FSM) state or SNA control vector 45, subfield 80 depending on resource type.

Operations

GET, SNAPSHOT(all types), NOTIFICATIONS

Attribute of

all objects except luGroup

cdrscRealLUname

Syntax

SNAcsAD-819(SIZE(0..17))

Meaning

Represents the network-qualified real LU name (instead of an alias name) for a cross-domain resource.

Source

Valid for a cross-domain resource that has been verified by session establishment with the actual resource represented by the CDRSC. The real name may vary from the CDRSC name due to alias name translation.

Operations
GET, SNAPSHOT (luCollection, luIndex)
Attribute of
crossDomainResource

connectionID

Syntax
GraphicString
Meaning
Address or addresses used to access the port
Source
The information provided is similar to that provided for adapterNumbers. However, this attribute provides the data in character format that might contain hexadecimal characters.

- For a port object, the value is described by the following:

  Channel lines
  This is the *hex* representation of the channel unit address coded on the ADDRESS operand of the LINE definition statement.

  Multipath channel
  This is the *hex* representation of each read and write channel unit address coded on the READ and WRITE operands of the LINE definition statement in the MPC group. Each address each separated by a comma.

  APPN host-to-host channel
  This is the *hex* representation of each read and write channel unit address coded on the READ and WRITE operands of the transport resource list entry (TRLE) associated with the PU. Each address is separated by a comma.

  NCP SDLC lines
  This is the *decimal* representation of the line address coded on ADDRESS operand of the LINE definition statement.

  LAN or ATM LAN emulation lines
  The *hex* channel unit address of the channel that connects VTAM to the IBM 3172 Nways® Interconnect Controller or the IBM S/390 Open Systems Adapter. This attribute appends the slot number to the channel address separated by a period; for example, `590.001`. The slot number is coded on the ADAPTNO operand of the PORT statement in the external communications adapter (XCA) major node.

  ATM native SVCs (switched lines) and PVCs (nonswitched lines)
  The name of the TRLE definition statement in the TRL major node that defines the IBM S/390 Open Systems Adapter.

  XCA Lines for Enterprise Extender
  This is the local host virtual IP address (VIPA). The IP address can be either an IPv4 address in dotted-decimal form or an IPv6 address in the colon-hexadecimal form.

- For a logicalLink object, the value is described by the following:

  ATM native SVCs (switched lines) and PVCs (nonswitched lines)
  The virtual path connection identifier/virtual channel identifier (VPCI/VCI) received on the CM_CONNECT indication.

Switched PUs for Enterprise Extender lines
This is the remote host virtual IP address (VIPA). The IP address can be either an IPv4 address in dotted-decimal form or an IPv6 address in the colon-hexadecimal form.

Operations
GET, SNAPSHOT(snaLocalTopo-appnPlusSubarea)

Attribute of
logicalLink
port

connectionType

Syntax
ENUMERATED { unknown,

host,
peer,
host-and-peer }

Meaning
Type of connection to node:
peer T2.1 nodes not requesting ACTPU
host-and-peer
T2.1 nodes requesting ACTPU
host FID4 connections
unknown
Inactive FID2 connections

Source
For type 2.1 node, the XID format 3 indicates ACTPU requirements. FID4 connections are determined by system definition.

Operations
GET

Attribute of
logicalLink

cp-cpSessionSupport

Syntax
BOOLEAN
TRUE APPN TG is capable of CP-CP sessions.
FALSE
APPN TG is not capable of CP-CP sessions.

Meaning
Whether TG is capable of supporting CP-CP sessions. This does NOT indicate whether CP-CP sessions exist; it indicates only that the capability exists.

Source
Determined from TG control vector X'47'.

Operations
SNAPSHOT(snaNetwork), SNAPSHOT(snaLocalTopo)

Attribute of
appnTransmissionGroup

definitionGroupName

Syntax
GraphicString

Meaning

Major node type and major node name.

Source

Major node type and name are determined from system definition. The type and name are concatenated and separated by a period; for example, `NCP.NCP3AB7`.

The following major node types are supported.

Prefix	Description
NCP	NCP major node
APPL	Application major node
LCLNONSNA	Local non-SNA major node
SWITCHED	Switched major node
LOCALSNA	Local SNA major node
CDRM	CDRM major node
CDRSC	CDRSC major node
CA	Channel Attached major node
MODEL	Model major node
LAN	ICA LAN major node
PACKET	Packet major node
XCA	XCA major node
LUGROUP	LUGROUP major node
ADJCP	Adjacent CP major node
TCP	TCP/IP major node
TRL	Transport resource list major node

Operations

GET, SNAPSHOT(snaLocalTopo)

Attribute of

definitionGroup

dependencies

Syntax

```
CHOICE { unknown -0- IMPLICIT NULL,

noDependents -1- IMPLICIT NULL,
dependendents Dependents }

Dependents ::= CHOICE { item ObjectInstance,

and IMPLICIT SET OF Dependents,
or IMPLICIT SET OF Dependents}
```

Meaning

Higher level object upon which the object of interest is dependent. This usually includes the definitionGroup object that has information about the VTAM major node and type.

Object type
Dependent on

port definition group, logicalLink, VTAM, NCP

NTRI logical lines that are represented as port objects report the physical unit as a dependency, when known.

logicalLink
definition group, port

Switched logicalLinks report a port dependency only when the PU is connected.

Note: VTAMTOPO line filtering does not affect port dependency.

t4Node
definition group, VTAM

logical unit
definition group, logicalLink, VTAM, dependent LU requester

Dependent logical units have a dependency on the owning PU. The owning PU is represented as a logicalLink. The logicalLink is not included for logicalUnits owned by the VTAM host, such as application programs.

Dependent LUs that use the dependent LU requester and dependent LU server function are dependent on the dependent LU requester node, which is represented as a snaNode.

cdrsc definition group, snaNode

CDRSCs have a dependency on the owning CDRM, when known. The owning CDRM is represented as an snaNode.

Source
The major node type and name and the higher level resource name are determined from system definitions.

Operations
GET, SNAPSHOT(snaLocalTopo, snaNetwork, luCollection)

Attribute of
all objects except luGroup

dlcName

Syntax
GraphicString (SIZE(1..8))

Meaning
A character constant describing the data link control name, as shown in the following list. VTAM might not support each DLC listed.

Constant
Description

IBMTRNET
Token Ring

FDDI Fiber

SDLC SDLC

CSMA
CSMA

FRRELAY
Frame Relay

SMDS
SMDS
CHANNEL
Channel
ETHERAND
Ethernet
TOKENBUS
Token Bus
ISDNBASC
ISDN basic
ISDNPRI
ISDN primary
ISDNBB
ISDN broadband
ATM Asynchronous Transfer Mode
XCF Cross-system Coupling Facility

Source
Determined from system definition.

Operations
GET, SNAPSHOT (snaLocalTopo)

Attribute of
port

dlurList

Syntax
SET OF ObjectInstance

Meaning
The list of dependent LU requester (DLUR) nodes served by this dependent LU server (DLUS) node. VTAM always returns empty set.

Source
Not supported.

Operations
GET

Attribute of
appnNN
interchangeNode

dlurLocalLsAddress

Syntax
CHOICE { noLSaddr NULL, lsAddr OCTET STRING }

Meaning
The local DLUR DLC address.

Source
The MAC/SAP address of the DLUR LAN adapter used for the connection to the PU reporting this attribute. This value is in the form 11223344556601 where the first 12 characters are the MAC address and the last two are the SAP address. This information is available when the PU is active and the x'57' DLC address vector has been received.

Operations
GET, SNAPSHOT (snaLocalTopo)

Attribute of
logicalLink

When a DLUR supports downstream PUs, an instance with this behavior reports the local addressing information (for example, a LAN MAC and SAP at the DLUr's end) for the logical link between the DLUR and the downstream PU.

dlurName

Syntax
CHOICE { noInfo NULL, object ObjectInstance }

Meaning
The network-qualified name of the dependent LU requester (DLUR) node associated with this logicalLink.

Source
This attribute value is determined when a switched PU connects to a dependent LU requester.

Operations
GET, SNAPSHOT (snaLocalTopo)

Attribute of
logicalLink

endpointForArc

Syntax
CHOICE { noinfo NULL, object ObjectInstance }

Meaning
VTAM always returns noinfo NULL.

Source
Not supported.

Operations
GET

Attribute of
logicalLink
port

erList

Syntax
SET OF ObjectInstance

Meaning
VTAM always returns empty set.

Source
Not supported.

Operations
GET

Attribute of
interchangeNode
migrationDataHost
t4Node
t5Node

extendedAppnNodeCapabilities

Syntax
OCTET STRING (SIZE(2))

Meaning
SNA control vector 45, subfield 81:

Bit	Meaning

1... .... Node is central director server

.000 0000
Reserved

0000 0000
Reserved

Source
The VTAM topology data base.

Operations
GET, SNAPSHOT (snaNetwork, snaLocalTopo)

Attribute of
interchangeNode
appnNN

functionID

Syntax
CHOICE { number INTEGER, string GraphicString }

Meaning
Value of the low-order relative distinguished name in the distinguished name of the object. This is the common name of the object.

Source
The value is determined from the GET request.

Operations
GET

Attribute of
all objects except luGroup

gatewayNode

Syntax
BOOLEAN
TRUE The type 4 node is capable of acting as a gateway node.
FALSE
The type 4 node is not capable of acting as a gateway node.

Meaning
Whether type 4 node is capable of acting as a gateway node.

Source
This capability is indicated on the ACTPU response.

Operations
GET, SNAPSHOT (snaLocalTopo)

Attribute of
t4Node

gatewaySSCP

Syntax
BOOLEAN
TRUE The node running the VTAM topology agent is capable of acting as a gateway node
FALSE
The node running the VTAM topology agent is not capable of acting as a gateway node

Meaning
Whether the node running the VTAM topology agent is capable of acting as a gateway node

Source
Start option definition for GWSSCP start option.
Operations
GET, SNAPSHOT(snaLocalTopo)
Attribute of
interchangeNode
migrationDataHost
t5Node

interconnectedNetids

Syntax
SET OF SEQUENCE { native BOOLEAN,

netid ObjectInstance,
netIdRole ENUMERATED { static (0),

dynamic (1) },

subareaAddress subareaAddress,
subareaLimit SubareaLimit }
Meaning
The network identifiers supported by a gateway NCP.
Source
These network identifiers are either defined on the NETWORK operand of the NCP major node or they are discovered by using the NOTIFY RUs from the NCP.
Operations
GET, SNAPSHOT(snaLocalTopo)
Attribute of
t4Node

limitedResource

Syntax
BOOLEAN
TRUE The port is a limited resource.
FALSE
The port is not a limited resource.
Meaning
Whether the port is a limited resource.
Source
System defintion for line represented by port object. Limited resource status is indicated by the LIMRES keyword.
Operations
GET
Attribute of
port

limitedResourceTimeout

Syntax
CHOICE { integer INTEGER,uninitialized NULL }
Meaning
This attribute is always returned as uninitialized NULL.
Source
Not supported.

Operations
GET
Attribute of
port

lineType

Syntax
ENUMERATED { switched, nonswitched }
Meaning
Depends on whether the line is switched.
switched
The line or physical unit is a switched resource.
nonswitched
The line or physical unit is not a switched resource.

For ATM native connections: are nonswitched.
Source
System definition for the PU.
Operations
GET, SNAPSHOT (snaLocalTopo)
Attribute of
logicalLink
port

linkName

Syntax
GraphicString (SIZE(1..17))
Meaning
The name of the physical unit represented by the logicalLink object.
Source
System definition for the PU or link station.
Operations
GET
Attribute of
logicalLink

linkStationRole

Syntax
ENUMERATED { secondary,

primary,
negotiable,
unknown }
Meaning
Indicates the role of the link station represented by the logicalLink object.
Source
This is determined by system definition and XIDs where applicable.
Operations
GET
Attribute of
logicalLink
port

luGroupMembers

Syntax

SET OF Fully-QualifiedNAUname
SNAcsAD-819(SIZE(1..17))

Meaning

Network-qualified names of the resources that make up an luGroup object.

Source

Depends on the underlying implementation of the luGroup.

USERVAR

The name specified on the VALUE= keyword when defining a USERVAR.

Generic resource

The names of real application programs associated with the generic resource definition. Application programs are added to or deleted from the luGroup with the SETLOGON macroinstruction of the VTAM application programming interface (API).

Operations

GET, SNAPSHOT (luCollection)

Attribute of

luGroup

luGroupName

Syntax

SNAcsAD-819(SIZE(1..8))

Meaning

The naming attribute of the luGroup object.

Source

This is the name provided on a GET request for an luGroup object.

Operations

GET

Attribute of

luGroup

luGroupSize

Syntax

NonnegativeNumber

Meaning

Number of members in the luGroup object.

Source

For a USERVAR, this is always 1. For a generic resource, this is the number of application programs associated with the generic resource. This corresponds to the number of member names provided in the luGroupMembers attribute.

Operations

GET

Attribute of

luGroup

luSecondName

Syntax

GraphicString

Meaning
For logicalUnit objects that are VTAM application programs, this attribute provides the ACB name of the application. For logicalUnit objects that are not application programs, this attribute value is the null string.

Source
The ACBNAME is coded on the APPL application definition in an application program major node. The ACBNAME may be the same as the application name.

Operations
GET

Attribute of
logicalUnit

maxBTUsize

Syntax
CHOICE { maxBTUsize INTEGER (1..32767), noMaxBTUsize NULL }

Meaning
VTAM always builds (noMaxBTUsize NULL).

Source
This attribute value is not supported.

Operations
GET

Attribute of
logicalLink
port

nameBinding

Syntax
OBJECT IDENTIFIER

Meaning
The name binding object from which VTAM derives the naming attribute of the object class.

Source
Constant for each object class

Operations
GET

Attribute of
all objects

nativeStatus

Syntax
INTEGER { active (0),

activeWithSession (1),
inactive (2),
neverActive (3),
pendingActive (4),
pendingInactive (5),
connectable (6),
routable (7),
operative (8),
congested (9),

released (10),
reset (11),
inoperative (12) }

OCTET: X'02'= inactive

X'03'= neverActive
X'04'= pendingActive
X'05'= pendingInactive
X'06'= connectable
X'07'= routable
X'09'= congested
X'0A'= released
X'0B'= reset
X'0C'= inoperative
X'FF'= no change

INTEGER is used for GET and NOTIFICATION operations.

OCTET is used for snapshot operations.

Meaning

The VTAM status of the resource. All of the states except the "congested" status correspond with existing VTAM resource states. The "congested" status indicates that an NCP type 4 node is in slowdown.

Source

The finite state machine and modifiers in the RDTE for the resource.

Note: This value usually corresponds with the resource status displayed on VTAM message IST486I. However, the VTAM display is usually more specific since VTAM defines many more intermediate resource states than are provided by the VTAM topology agent.

Operations

GET, SNAPSHOT (all types), NOTIFICATIONS

Attribute of

all objects except luGroup

nlrResidentNodePointer

Syntax

GraphicString

Meaning

Name of the CP or SSCP that owns the real resource represented by this CDRSC or appnRegisteredLu object.

Source

Depends on Object type

CDRSC

This is the owning CP or SSCP. This may be predefined or learned as a result of session establishment. When this information is unknown, the null string is returned.

appnRegisteredLu

This is the CP name for this resource.

Operations

GET, SNAPSHOT (luCollection)

Attribute of
CDRSC
appnRegisteredLu

nnServerPointer

Syntax
CHOICE { noObject NULL, Object ObjectInstance }
Meaning
This is the object instance that represents the network node server for the node running the VTAM topology agent when the node is an APPN end node or migration data host. The null form is provided when the network node server is not known.
Source
The network node server is determined at the time CP-CP sessions are established. Potential network node servers might be defined in a network node server list major node, but this attribute is available only when a server has actually been selected.
Operations
GET
Attribute of
appnEN
migrationDataHost

nonLocalResourceName

Syntax
GraphicString
Meaning
The name specified on the GET operation for the registeredLU object or the CDRSC object.
Source
The input name is returned in this attribute.
Operations
GET
Attribute of
CDRSC
appnRegisteredLu

nonLocalResourceType

Syntax
GraphicString
Meaning
The object type of the object found with the nonLocalResourceName attribute. The values are:
- CDRSC
- appnRegisteredLu.

Source
The resulting type depends on the type of the resource found as a result of a GET operation. If both exists, CDRSC is returned.
Operations
GET
Attribute of
CDRSC
appnRegisteredLu

objectClass

Syntax

OBJECT IDENTIFIER

Meaning

The object class of the object containing this attribute

Source

From VTAM resource information

Operations

GET

Attribute of

all objects

opEquipmentList

Syntax

SET OF ObjectInstance

Meaning

For the node running the VTAM topology agent, a distinguished name in the form:

```
distinguishedName
"1.3.18.0.2.4.8=ORGREG;
 2.5.4.10=IBM;
 1.3.18.0.2.4.7=<CPU Model>;
 1.3.14.2.2.4.50=<CPU Serial Number>"
```

<CPU Model> is a character string that contains the actual CPU model for the agent host. <CPU Serial Number> is a character string that contains the serial number for the agent host.

For non-host objects, the null set is returned.

Source

Determined from system storage.

Operations

GET, SNAPSHOT (snaLocalTopo)

Attribute of

appnEN
interchangeNode
lenNode
logicalLink
migrationDataHost
appnNN
port
t2-1Node
t4Node
t5Node

opNetworkName

Syntax

GraphicString

Meaning

The network name of the network where the LU represented by the CDRSC object resides, when available.

Source

For predefined alias CDRSCs that are not in session, this information is

unknown and the null string is provided. Otherwise, this is the predefined or learned NETID of the resource represented by this CDRSC object.

Operations

GET

Attribute of

CDRSC

operationalState

Syntax

ENUMERATED { disabled, enabled } OCTET: X'00'= disabled

X'01'= enabled
X'FF'= no change

ENUMERATED is used for GET and NOTIFICATION operations.

OCTET is used for snapshot operations.

Meaning

The OSI operational state.

Source

Determined from VTAM resource definition table entry (RDTE) finite state machine (FSM) state or SNA control vector 45, subfield 80 depending on resource type.

Operations

GET, SNAPSHOT (all types), NOTIFICATIONS

Attribute of

all objects except luGroup

packages

Syntax

SET OF OBJECT IDENTIFIER

Meaning

The packages of attributes supported by VTAM for an object class.

Source

Constant set of attributes for each class

Operations

GET

Attribute of

all objects

partnerConnection

Syntax

CHOICE { noinfo NULL,object ObjectInstance }

Meaning

This is the object instance that represents the logicalLink on other end of a subarea or APPN TG connection, when available.

Source

This is the name of the partner link station provided by the X'0EF7' control vector on XID. This information is available from APPN nodes and uplevel subarea nodes. Uplevel subarea nodes are VTAM V4R3 or NCP V7R1 or later. LogicalLinks must be active to obtain this information.

Operations

GET

Attribute of
logicalLink
port

portId

Syntax
GraphicString
Meaning
Name of the port object. When reported as an attribute of logicalLink, this identifies the SNA line associated with the physical unit. When reported as an attribute of port, it names the port object.
Source
The port object represents a SNA LINE or DAN. SNA lines are defined during system definition or are dynamically created when channel attached NCPs are activated. Dynamically created lines have names of the form `0321-L` where the first 4 characters are the printable hex representation of the channel unit address coded on the CUADDR operand of the PCCU statement in the NCP major node.

DANs represent connections to SNA controllers and are not explicitly defined. The name for the DAN is constructed from the channel unit address on the CUADDR operand of the PU statement in the local SNA major node. For example, CUADDR=16, would result in a portid of `000016-L`.
Operations
GET, SNAPSHOT (snaLocalTopo)
Attribute of
logicalLink
port

proceduralStatus

Syntax
SET OF INTEGER { initializationRequired (0)

notInitialized (1)
initializing (2)
reporting (3)
terminating (4) }

OCTET: X'00'= no status

X'08'= terminating
X'10'= reporting
X'20'= initializing
X'40'= not initialized

INTEGER is used for GET and NOTIFICATION operations.

OCTET is used for snapshot operations.
Meaning
OSI state procedural status.
Source
Determined from VTAM resource definition table entry (RDTE) finite state machine (FSM) state or SNA control vector 45, subfield 80 depending on resource type.

Operations
GET, SNAPSHOT (all types), NOTIFICATIONS
Attribute of
all objects except luGroup and definitionGroup

puName

SNAcsA-819 (SIZE(1..8))
Meaning
This is the name of the VTAM Agent host's subarea PU.
Source
The VTAM host subarea PU name is defined on the HOSTPU start option or is defaulted to `ISTPUS`.
Operations
GET, SNAPSHOT (snaLocalTopo)
Attribute of
interchangeNode
migrationDataHost
t5Node

receiveWindowSize

Syntax
CHOICE { integer INTEGER, uninitialized NULL }
Meaning
VTAM always returns uninitialized NULL.
Source
Not supported
Operations
GET
Attribute of
port

realSSCPname

Syntax
SNAcsAD-819 (SIZE(0..17))
Meaning
The real name of a cross domain resource manager as known at its SSCP. This information is not known until a CDRM-CDRM session has been established to the CDRM.
Source
This information is determined from SSCP Name control vector X'18' when available.
Operations
SNAPSHOT (snaNetwork-appnPlusSubarea)
Attribute of
crossDomainResourceManager

registeredBy

Syntax
ObjectInstance
Meaning
Provides the name of the node which registered the logical unit represented by this nonlocal resource.

Source

Name of the end node that registered the LU. This attribute is only known at Directory Server and Network Node Server agent hosts that have the resource registered.

Operations

GET

Attribute of

appnRegisteredLu

relatedAdapter

Syntax

CHOICE { noinfo NULL, object ObjectInstance }

Meaning

A logicalLink instance that is providing the physical connection for a logical line represented by a port object.

Source

The physical resource is applicable to NTRI logical lines and is determined at connection time. The physical PU is determined by the Related Resource Network Name subfield of the X'57' control vector provided by NCP.

Operations

GET, SNAPSHOT (snaLocalTopo)

Attribute of

port

residentNodePointer

Syntax

ObjectInstance

Meaning

Name of a managed object representing the SNA node upon which this logicalUnit resides.

Source

For Dependent LUs, this represents the PU under which the LU is defined. For application programs and local non-SNA terminals, this represents the VTAM host.

Operations

GET, SNAPSHOT (luCollection)

Attribute of

logicalUnit

resourceSequenceNumber

Syntax

INTEGER (0..2**32-1)

Meaning

For a GET on the VTAM agent host object, this attribute provides the current resource sequence number for the node. For SNAPSHOT(snaNetwork) this attribute provides the current resource sequence number for an appnTransmissionGroup object.

Source

VTAM resource data

Operations

GET, SNAPSHOT (snaNetwork)

Attribute of
interchangeNode
appnNN
appnTransmissionGroup

routeAdditionResistance

Syntax
INTEGER (0..255)
Meaning
VTAM always provides the value 0.
Source
Not supported.
Operations
GET
Attribute of
interchangeNode
appnNN

sendWindowSize

Syntax
CHOICE { integer INTEGER, uninitialized NULL }
Meaning
VTAM always provides uninitialized NULL.
Source
Not supported.
Operations
GET
Attribute of
port

snaNodeName

Syntax
SNAcsAD-819 (SIZE(1..17))
Meaning
The name of an SNA node.
Source
For the VTAM host objects, this is the CP or SSCP name. For t4Node objects, this is the NCP PU name. For t2-1Node or lenNode, this is the CP name.

Note: GET support for all node types except t4Node is limited to the VTAM agent host.
Operations
GET
Attribute of
appnEN
interchangeNode
lenNode
migrationDataHost
appnNN
t2-1Node
t4Node
t5Node

softwareList

Syntax

SET OF ObjectInstance

Meaning

Provides the version and release of the VTAM running at the agent host.

```
distinguishedName
"1.3.18.0.2.4.8=ORGREG;
 2.5.4.10=IBM;
 0.0.13.3100.0=(pString
 ACF/VTAM.<version>.<release>.
          <dot rel>)"
```

Source

VTAM storage.

Operations

GET, SNAPSHOT (snaLocalTopo)

Attribute of

appnEN
interchangeNode
lenNode
migrationDataHost
appnNN
t2-1Node
t5Node

subareaAddress

Syntax

INTEGER (1..65535)

Meaning

The subarea address associated with the subarea object instance.

Source

For the agent host object types, this is the value of the HOSTSA start option.

For t4Node objects that represent NCPs, this is the value of the SUBAREA operand of the PCCU statement in an NCP major node.

Operations

GET, SNAPSHOT (snaLocalTopo)

Attribute of

interchangeNode
migrationDataHost
t4Node
t5Node

subareaLimit

Syntax

INTEGER (255..65535)

Meaning

The subarea limit associated with the subarea object instance.

Source

For the agent host object types, this is the value of the MXSUBNUM start option.

For t4Node objects that represent NCPs, this is the value of the SALIMIT operand of the NETWORK statement in an NCP major node. VTAM obtains this value at ACTPU response time, not from the NCP definition.

Operations
GET, SNAPSHOT (snaLocalTopo)
Attribute of
interchangeNode
migrationDataHost
t4Node
t5Node

supportedResources

Syntax
CHOICE { noResources NoResources, resources SET OF ObjectInstance }

NoResources ::= ENUMERATED { infoUnavailable,

none }
Meaning
When provided for the VTAM agent host object, this attribute contains a set of definitionGroup objects that represent all the major nodes defined at this host. For all other object types, (`noResources none`) is returned.
Source
This information is obtained from the system definitions at the agent host.
Operations
GET
Attribute of
all objects except luGroup

sysplexInfo

Syntax
GraphicString
Meaning
Name of the MVS/ESA sysplex, if known.
Source
This name is obtained from the MVS/ESA CVT when available.
Operations
GET, SNAPSHOT (snaLocalTopo)
Attribute of
appnEN
interchangeNode
lenNode
migrationDataHost
appnNN
t2-1Node
t5Node

tn3270ClientDnsName

Syntax
CHOICE { noDnsName NULL, fullName GraphicString, truncatedName GraphicString }
Meaning
The TN3270 client DNS name.
Source
The client DNS name associated with TN3270 LU.

Operations
GET, SNAPSHOT (luCollection), NOTIFICATIONS
Attribute of
logicalUnit

This attribute returns either a GraphicString representation of a Domain Name Service (DNS) name for the TN3270 client associated with this LU, or a NULL value indicating that there is no client DNS name associated with this LU. If a name is returned, there is an indication whether it is a full name or a truncated name.

tn3270ClientIpAddress

Syntax
CHOICE { noIpAddress NULL, ipv4 GraphicString(SIZE(7..15)), ipv6 GraphicString(SIZE(2..56))}
Meaning
The TN3270 client IP address.
Source
The IP address associated with TN3270 LU.
Operations
GET, SNAPSHOT (luCollection), NOTIFICATIONS
Attribute of
logicalUnit

This attribute returns either a NULL value, indicating that there is no client IP address associated with this LU, or an IPv4 address in the dotted–decimal form (for example, a.b.c.d), or an IPv6 address in the colon–hexadecimal form (for example, a:b:c:d:e:f:g:h) or if the IPv6 zone ID is supported, displays could be in the form a:b:c:d:e:f:g:h%zoneid.

tn3270ClientPortNumber

Syntax
CHOICE { noIpPort NULL, portNumber PrintableString(SIZE(1..5)
Meaning
The TN3270 client IP port number.
Source
The IP port number associated with TN3270 LU.
Operations
GET, SNAPSHOT (luCollection), NOTIFICATIONS
Attribute of
logicalUnit

This attribute returns either a PrintableString representation of an IP port number between 1 and 65535 (decimal) inclusive, or a NULL value indicating that there is no client IP port number associated with this LU.

transmissionGroupNumber

Syntax
CHOICE { integer INTEGER, uninitialized NULL }
Meaning
This attribute provides the TG number associated with the connection provided by this logicalLink. If the TG number is unknown, `(uninitialized NULL)` is provided.
Source
The TG number may be predefined with the TGN operand of the PU

statement in system definition or may be dynamically assigned if not predefined. Not all PUs represented as logicalLinks will have TG numbers (for example, PU T1). PUs or link stations with the TG defined as ANY, where the TG number has not been negotiated, will return uninitialized NULL.

Operations

GET

Attribute of

logicalLink

underlyingConnectionNames

Syntax

SET OF ObjectInstance

Meaning

For a port object, this is always the null set.

For a logicalLink object, this represents the port that the logicalLink is subordinate to. If unknown, the null set is provided.

Source

For non-switched PUs, this is always the port object representing the LINE that the physical unit is defined under. For switched PUs, this attribute will not be known if the PU is not dialed.

Operations

GET

Attribute of

logicalLink
port

userLabel

Syntax

GraphicString

Meaning

For logicalUnit objects that represent VTAM applications, this attribute contains the name of the application ACB. For CDRSC objects, this attribute contains the name of the CDRSC LUALIAS name if coded. For all other logicalUnit and CDRSC objects this attribute is the null string.

Source

VTAM applications define the ACB name on the ACBNAME operand of the APPL statement in a VTAM application program major node. The CDRSC LUALIAS name is coded on the LUALIAS operand of the CDRSC statement for a predefined CDRSC.

Operations

GET, SNAPSHOT (luCollection)

Attribute of

CDRSC
logicalUnit

unknownStatus

Syntax

BOOLEAN
TRUE Unknown status
FALSE
Status is known

OCTET: X'00'= FALSE

X'01'= TRUE
X'FF'= no change

BOOLEAN is used for GET and NOTIFICATION operations.

OCTET is used for snapshot operations.

Meaning
OSI state unknown status.

Source
Determined from VTAM resource definition table entry (RDTE) finite state machine (FSM) state or SNA control vector 45, subfield 80 depending on resource type.

Operations
GET, SNAPSHOT (all types), NOTIFICATIONS

Attribute of
all objects except luGroup

usageState

Syntax
ENUMERATED { idle,

active,
busy }

OCTET: X'00'= idle

X'01'= active
X'02'= busy
X'FF'= no change

Meaning
OSI state usage state

Source
Determined from VTAM resource definition table entry (RDTE) finite state machine (FSM) state or SNA control vector 45, subfield 80 depending on resource type.

Operations
GET

Attribute of
appnNN
port
appnRegisteredLu
t2-1Node
t4Node
t5Node

Appendix G. VTAMTOPO filtering option reporting

Table 42 on page 347 summarizes the results of using the VTAMTOPO filtering option for reporting a switched PU under an NCP.

The following is the legend for the table:

notIGNR/INCL
Neither IGNORE or INCLUDE specified.

r VTAMTOPO= not specified, REPORT inherited from node above.

R VTAMTOPO=REPORT specified (or NOLLINES at containing GROUP).

nr VTAMTOPO= not specified, NOREPORT inherited from node above.

NR VTAMTOPO=NOREPORT specified (or NOSWPUS at containing GROUP).

NotRep
Switched PU is not reported.

IGNR VTAMTOPO=IGNORE specified at designated major node.

INCL VTAMTOPO=INCLUDE specified at the designated major node.

any Value of VTAMTOPO= does not matter, inclusion not specified.

R-NCP
Switched PU is reported under the NCP under which the Switched PU is connected.

R-SSCP
Switched PU is reported under the SSCP directly (not under any NCP), as it is when it is not connected.

Notes:

1. Values shown are assumed to be set in the applicable major node, individual PUX or SW PU before the connection is established. A MODIFY VTAMTOPO to set these values after the connection is established may not show the expected result before the connection is taken down and reestablished.
2. This table can also be used for the Switched PUs that are connected under an XCA. In the results column, all R-NCP will be R-SSCP, since the XCA is not represented by a PU type, and does not appear as a discrete node in SNA local topology.
3. The PUX is a place holder for a future connected switched PU under the switched line. It takes its VTAMTOPO value either explicitly from the line GROUP value, or implicitly from the NCP or the XCA major node value. If its VTAMTOPO value is explicitly set, it cannot be modified.

Table 42. Connected switched PU report

VTAMTOPO value on				SPWU result
NCP	PUX	SWND	SWPU	
notIGNR/INCL	r/R	R	r	R-NCP
notIGNR/INCL	r/R	INCL	r	R-NCP
notIGNR/INCL	r/R	R	R	R-NCP
notIGNR/INCL	r/R	INCL	R	R-NCP

Table 42. Connected switched PU report (continued)

VTAMTOPO value on				SPWU result
NCP	**PUX**	**SWND**	**SWPU**	
notIGNR/INCL	r/R	R	NR	NotRep
notIGNR/INCL	r/R	INCL	NR	R-NCP
notIGNR/INCL	r/R	NR	nr	R-NCP
notIGNR/INCL	r/R	IGNR	nr	R-NCP
notIGNR/INCL	r/R	NR	NR	NotRep
notIGNR/INCL	r/R	IGNR	NR	NotRep
notIGNR/INCL	r/R	NR	R	R-NCP
notIGNR/INCL	r/R	IGNR	R	R-NCP
notIGNR/INCL	nr/NR	R	r	R-SSCP
notIGNR/INCL	nr/NR	INCL	r	R-SSCP
notIGNR/INCL	nr/NR	R	R	R-NCP
notIGNR/INCL	nr/NR	INCL	R	R-SSCP
notIGNR/INCL	nr/NR	R	NR	NotRep
notIGNR/INCL	nr/NR	INCL	NR	R-SSCP
notIGNR/INCL	nr/NR	NR	nr	NotRep
notIGNR/INCL	nr/NR	IGNR	nr	NotRep
notIGNR/INCL	nr/NR	NR	NR	NotRep
notIGNR/INCL	nr/NR	IGNR	NR	NotRep
notIGNR/INCL	nr/NR	NR	R	R-NCP
notIGNR/INCL	nr/NR	IGNR	R	NotRep
INCL	any	R	r	R-NCP
INCL	any	INCL	r	R-NCP
INCL	any	R	R	R-NCP
INCL	any	INCL	R	R-NCP
INCL	any	R	NR	NotRep
INCL	any	INCL	NR	R-NCP
INCL	any	NR	nr	R-NCP
INCL	any	IGNR	nr	R-NCP
INCL	any	NR	NR	NotRep
INCL	any	IGNR	NR	NotRep
INCL	any	NR	R	R-NCP
INCL	any	IGNR	R	R-NCP
IGNR	any	R	r	R-SSCP
IGNR	any	INCL	r	R-SSCP
IGNR	any	R	R	R-SSCP
IGNR	any	INCL	R	R-SSCP
IGNR	any	R	NR	NotRep
IGNR	any	INCL	NR	R-SSCP
IGNR	any	NR	nr	NotRep

Table 42. Connected switched PU report (continued)

VTAMTOPO value on				SPWU result
NCP	**PUX**	**SWND**	**SWPU**	
IGNR	any	IGNR	nr	NotRep
IGNR	any	NotRep	NotRep	NotRep
IGNR	any	IGNR	NR	NotRep
IGNR	any	NR	R	R-SSCP
IGNR	any	IGNR	R	NotRep

Appendix H. Architectural specifications

This appendix lists documents that provide architectural specifications for the SNA Protocol.

The APPN Implementers' Workshop (AIW) architecture documentation includes the following architectural specifications for SNA APPN and HPR:

- APPN Architecture Reference (SG30-3422-04)
- APPN Branch Extender Architecture Reference Version 1.1
- APPN Dependent LU Requester Architecture Reference Version 1.5
- APPN Extended Border Node Architecture Reference Version 1.0
- APPN High Performance Routing Architecture Reference Version 4.0
- SNA Formats (GA27-3136-19)
- SNA Technical Overview (GC30-3073-04)

For more information, refer to the AIW documentation page at http://nhdidd.raleigh.ibm.com/app/aiwdoc.htm.

The following RFC also contains SNA architectural specifications:

- RFC 2353 *APPN/HPR in IP Networks APPN Implementers' Workshop Closed Pages Document*

RFCs can be obtained from:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

Many RFCs are available online. Hardcopies of all RFCs are available from the NIC, either individually or by subscription. Online copies are available using FTP from the NIC at http://www.rfc-editor.org/rfc.html.

Use FTP to download the files, using the following format:

```
RFC:RFC-INDEX.TXT
RFC:RFCnnnn.TXT
RFC:RFCnnnn.PS
```

where:

- *nnnn* is the RFC number.
- TXT is the text format.
- PS is the postscript format.

You can also request RFCs through electronic mail, from the automated NIC mail server, by sending a message to `service@nic.ddn.mil` with a subject line of `RFC nnnn` for text versions or a subject line of `RFC nnnn.PS` for PostScript versions. To request a copy of the RFC index, send a message with a subject line of `RFC INDEX`.

For more information, contact `nic@nic.ddn.mil`.

Appendix I. Related protocol specifications (RFCs)

This appendix lists the related protocol specifications for TCP/IP. The Internet Protocol suite is still evolving through requests for comments (RFC). New protocols are being designed and implemented by researchers and are brought to the attention of the Internet community in the form of RFCs. Some of these protocols are so useful that they become recommended protocols. That is, all future implementations for TCP/IP are recommended to implement these particular functions or protocols. These become the *de facto* standards, on which the TCP/IP protocol suite is built.

You can request RFCs through electronic mail, from the automated Network Information Center (NIC) mail server, by sending a message to `service@nic.ddn.mil` with a subject line of `RFC` *`nnnn`* for text versions or a subject line of `RFC` *`nnnn`*`.PS` for PostScript versions. To request a copy of the RFC index, send a message with a subject line of `RFC INDEX`.

For more information, contact `nic@nic.ddn.mil` or at:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

Hard copies of all RFCs are available from the NIC, either individually or by subscription. Online copies are available at the following Web address: http://www.rfc-editor.org/rfc.html.

See "Internet drafts" on page 366 for draft RFCs implemented in this and previous Communications Server releases.

Many features of TCP/IP Services are based on the following RFCs:

RFC	Title and Author
652	*Telnet output carriage-return disposition option* D. Crocker
653	*Telnet output horizontal tabstops option* D. Crocker
654	*Telnet output horizontal tab disposition option* D. Crocker
655	*Telnet output formfeed disposition option* D. Crocker
657	*Telnet output vertical tab disposition option* D. Crocker
658	*Telnet output linefeed disposition* D. Crocker
698	*Telnet extended ASCII option* T. Mock
726	*Remote Controlled Transmission and Echoing Telnet option* J. Postel, D. Crocker
727	*Telnet logout option* M.R. Crispin
732	*Telnet Data Entry Terminal option* J.D. Day
733	*Standard for the format of ARPA network text messages* D. Crocker, J. Vittal, K.T. Pogran, D.A. Henderson

734 *SUPDUP Protocol* M.R. Crispin

735 *Revised Telnet byte macro option* D. Crocker, R.H. Gumpertz

736 *Telnet SUPDUP option* M.R. Crispin

749 *Telnet SUPDUP—Output option* B. Greenberg

765 *File Transfer Protocol specification* J. Postel

768 *User Datagram Protocol* J. Postel

779 *Telnet send-location option* E. Killian

783 *TFTP Protocol (revision 2)* K.R. Sollins

791 *Internet Protocol* J. Postel

792 *Internet Control Message Protocol* J. Postel

793 *Transmission Control Protocol* J. Postel

820 *Assigned numbers* J. Postel

821 *Simple Mail Transfer Protocol* J. Postel

822 *Standard for the format of ARPA Internet text messages* D. Crocker

823 *DARPA Internet gateway* R. Hinden, A. Sheltzer

826 *Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware* D. Plummer

854 *Telnet Protocol Specification* J. Postel, J. Reynolds

855 *Telnet Option Specification* J. Postel, J. Reynolds

856 *Telnet Binary Transmission* J. Postel, J. Reynolds

857 *Telnet Echo Option* J. Postel, J. Reynolds

858 *Telnet Suppress Go Ahead Option* J. Postel, J. Reynolds

859 *Telnet Status Option* J. Postel, J. Reynolds

860 *Telnet Timing Mark Option* J. Postel, J. Reynolds

861 *Telnet Extended Options: List Option* J. Postel, J. Reynolds

862 *Echo Protocol* J. Postel

863 *Discard Protocol* J. Postel

864 *Character Generator Protocol* J. Postel

865 *Quote of the Day Protocol* J. Postel

868 *Time Protocol* J. Postel, K. Harrenstien

877 *Standard for the transmission of IP datagrams over public data networks* J.T. Korb

883 *Domain names: Implementation specification* P.V. Mockapetris

884 *Telnet terminal type option* M. Solomon, E. Wimmers

885 *Telnet end of record option* J. Postel

894 *Standard for the transmission of IP datagrams over Ethernet networks* C. Hornig

896 *Congestion control in IP/TCP internetworks* J. Nagle

903 *Reverse Address Resolution Protocol* R. Finlayson, T. Mann, J. Mogul, M. Theimer

904 *Exterior Gateway Protocol formal specification* D. Mills

919 *Broadcasting Internet Datagrams* J. Mogul

922 *Broadcasting Internet datagrams in the presence of subnets* J. Mogul

927 *TACACS user identification Telnet option* B.A. Anderson

933 *Output marking Telnet option* S. Silverman

946 *Telnet terminal location number option* R. Nedved

950 *Internet Standard Subnetting Procedure* J. Mogul, J. Postel

951 *Bootstrap Protocol* W.J. Croft, J. Gilmore

952 *DoD Internet host table specification* K. Harrenstien, M. Stahl, E. Feinler

959 *File Transfer Protocol* J. Postel, J.K. Reynolds

961 *Official ARPA-Internet protocols* J.K. Reynolds, J. Postel

974 *Mail routing and the domain system* C. Partridge

1001 *Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods* NetBios Working Group in the Defense Advanced Research Projects Agency, Internet Activities Board, End-to-End Services Task Force

1002 *Protocol Standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications* NetBios Working Group in the Defense Advanced Research Projects Agency, Internet Activities Board, End-to-End Services Task Force

1006 *ISO transport services on top of the TCP: Version 3* M.T. Rose, D.E. Cass

1009 *Requirements for Internet gateways* R. Braden, J. Postel

1011 *Official Internet protocols* J. Reynolds, J. Postel

1013 *X Window System Protocol, version 11: Alpha update April 1987* R. Scheifler

1014 *XDR: External Data Representation standard* Sun Microsystems

1027 *Using ARP to implement transparent subnet gateways* S. Carl-Mitchell, J. Quarterman

1032 *Domain administrators guide* M. Stahl

1033 *Domain administrators operations guide* M. Lottor

1034 *Domain names—concepts and facilities* P.V. Mockapetris

1035 *Domain names—implementation and specification* P.V. Mockapetris

1038 *Draft revised IP security option* M. St. Johns

1041 *Telnet 3270 regime option* Y. Rekhter

1042 *Standard for the transmission of IP datagrams over IEEE 802 networks* J. Postel, J. Reynolds

1043 *Telnet Data Entry Terminal option: DODIIS implementation* A. Yasuda, T. Thompson

1044 *Internet Protocol on Network System's HYPERchannel: Protocol specification* K. Hardwick, J. Lekashman

1053 *Telnet X.3 PAD option* S. Levy, T. Jacobson

1055 *Nonstandard for transmission of IP datagrams over serial lines: SLIP* J. Romkey

1057 *RPC: Remote Procedure Call Protocol Specification: Version 2* Sun Microsystems

1058 *Routing Information Protocol* C. Hedrick

1060 *Assigned numbers* J. Reynolds, J. Postel

1067 *Simple Network Management Protocol* J.D. Case, M. Fedor, M.L. Schoffstall, J. Davin

1071 *Computing the Internet checksum* R.T. Braden, D.A. Borman, C. Partridge

1072 *TCP extensions for long-delay paths* V. Jacobson, R.T. Braden

1073 *Telnet window size option* D. Waitzman

1079 *Telnet terminal speed option* C. Hedrick

1085 *ISO presentation services on top of TCP/IP based internets* M.T. Rose

1091 *Telnet terminal-type option* J. VanBokkelen

1094 *NFS: Network File System Protocol specification* Sun Microsystems

1096 *Telnet X display location option* G. Marcy

1101 *DNS encoding of network names and other types* P. Mockapetris

1112 *Host extensions for IP multicasting* S.E. Deering

1113 *Privacy enhancement for Internet electronic mail: Part I — message encipherment and authentication procedures* J. Linn

1118 *Hitchhikers Guide to the Internet* E. Krol

1122 *Requirements for Internet Hosts—Communication Layers* R. Braden, Ed.

1123 *Requirements for Internet Hosts—Application and Support* R. Braden, Ed.

1146 *TCP alternate checksum options* J. Zweig, C. Partridge

1155 *Structure and identification of management information for TCP/IP-based internets* M. Rose, K. McCloghrie

1156 *Management Information Base for network management of TCP/IP-based internets* K. McCloghrie, M. Rose

1157 *Simple Network Management Protocol (SNMP)* J. Case, M. Fedor, M. Schoffstall, J. Davin

1158 *Management Information Base for network management of TCP/IP-based internets: MIB-II* M. Rose

1166 *Internet numbers* S. Kirkpatrick, M.K. Stahl, M. Recker

1179 *Line printer daemon protocol* L. McLaughlin

1180 *TCP/IP tutorial* T. Socolofsky, C. Kale

1183 *New DNS RR Definitions* C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. Mockapetris

1184 *Telnet Linemode Option* D. Borman

1186 *MD4 Message Digest Algorithm* R.L. Rivest

1187 *Bulk Table Retrieval with the SNMP* M. Rose, K. McCloghrie, J. Davin

1188 *Proposed Standard for the Transmission of IP Datagrams over FDDI Networks* D. Katz

1190 *Experimental Internet Stream Protocol: Version 2 (ST-II)* C. Topolcic

1191 *Path MTU discovery* J. Mogul, S. Deering

1198 *FYI on the X window system* R. Scheifler

1207 *FYI on Questions and Answers: Answers to commonly asked "experienced Internet user" questions* G. Malkin, A. Marine, J. Reynolds

1208 *Glossary of networking terms* O. Jacobsen, D. Lynch

1213 *Management Information Base for Network Management of TCP/IP-based internets: MIB-II* K. McCloghrie, M.T. Rose

1215 *Convention for defining traps for use with the SNMP* M. Rose

1227 *SNMP MUX protocol and MIB* M.T. Rose

1228 *SNMP-DPI: Simple Network Management Protocol Distributed Program Interface* G. Carpenter, B. Wijnen

1229 *Extensions to the generic-interface MIB* K. McCloghrie

1230 *IEEE 802.4 Token Bus MIB* K. McCloghrie, R. Fox

1231 *IEEE 802.5 Token Ring MIB* K. McCloghrie, R. Fox, E. Decker

1236 *IP to X.121 address mapping for DDN* L. Morales, P. Hasse

1256 *ICMP Router Discovery Messages* S. Deering, Ed.

1267 *Border Gateway Protocol 3 (BGP-3)* K. Lougheed, Y. Rekhter

1268 *Application of the Border Gateway Protocol in the Internet* Y. Rekhter, P. Gross

1269 *Definitions of Managed Objects for the Border Gateway Protocol: Version 3* S. Willis, J. Burruss

1270 *SNMP Communications Services* F. Kastenholz, ed.

1285 *FDDI Management Information Base* J. Case

1315 *Management Information Base for Frame Relay DTEs* C. Brown, F. Baker, C. Carvalho

1321 *The MD5 Message-Digest Algorithm* R. Rivest

1323 *TCP Extensions for High Performance* V. Jacobson, R. Braden, D. Borman

1325 *FYI on Questions and Answers: Answers to Commonly Asked "New Internet User" Questions* G. Malkin, A. Marine

1327 *Mapping between X.400 (1988)/ISO 10021 and RFC 822* S. Hardcastle-Kille

1340 *Assigned Numbers* J. Reynolds, J. Postel

1344 *Implications of MIME for Internet Mail Gateways* N. Bornstein

1349 *Type of Service in the Internet Protocol Suite* P. Almquist

1350 *The TFTP Protocol (Revision 2)* K.R. Sollins

1351 *SNMP Administrative Model* J. Davin, J. Galvin, K. McCloghrie

1352 *SNMP Security Protocols* J. Galvin, K. McCloghrie, J. Davin

1353 *Definitions of Managed Objects for Administration of SNMP Parties* K. McCloghrie, J. Davin, J. Galvin

1354 *IP Forwarding Table MIB* F. Baker

1356 *Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode* A. Malis, D. Robinson, R. Ullmann

1358 *Charter of the Internet Architecture Board (IAB)* L. Chapin

1363 *A Proposed Flow Specification* C. Partridge

1368 *Definition of Managed Objects for IEEE 802.3 Repeater Devices* D. McMaster, K. McCloghrie

1372 *Telnet Remote Flow Control Option* C. L. Hedrick, D. Borman

1374 *IP and ARP on HIPPI* J. Renwick, A. Nicholson

1381 *SNMP MIB Extension for X.25 LAPB* D. Throop, F. Baker

1382 *SNMP MIB Extension for the X.25 Packet Layer* D. Throop

1387 *RIP Version 2 Protocol Analysis* G. Malkin

1388 *RIP Version 2 Carrying Additional Information* G. Malkin

1389 *RIP Version 2 MIB Extensions* G. Malkin, F. Baker

1390 *Transmission of IP and ARP over FDDI Networks* D. Katz

1393 *Traceroute Using an IP Option* G. Malkin

1398 *Definitions of Managed Objects for the Ethernet-Like Interface Types* F. Kastenholz

1408 *Telnet Environment Option* D. Borman, Ed.

1413 *Identification Protocol* M. St. Johns

1416 *Telnet Authentication Option* D. Borman, ed.

1420 *SNMP over IPX* S. Bostock

1428 *Transition of Internet Mail from Just-Send-8 to 8bit-SMTP/MIME* G. Vaudreuil

1442 *Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1443 *Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1445 *Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Galvin, K. McCloghrie

1447 *Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)* K. McCloghrie, J. Galvin

1448 *Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1464 *Using the Domain Name System to Store Arbitrary String Attributes* R. Rosenbaum

1469 *IP Multicast over Token-Ring Local Area Networks* T. Pusateri

1483 *Multiprotocol Encapsulation over ATM Adaptation Layer 5* Juha Heinanen

1497 *BOOTP Vendor Information Extensions* J. Reynolds

1514 *Host Resources MIB* P. Grillo, S. Waldbusser

1516 *Definitions of Managed Objects for IEEE 802.3 Repeater Devices* D. McMaster, K. McCloghrie

1521 *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies* N. Borenstein, N. Freed

1533 *DHCP Options and BOOTP Vendor Extensions* S. Alexander, R. Droms

1534 *Interoperation Between DHCP and BOOTP* R. Droms

1535 *A Security Problem and Proposed Correction With Widely Deployed DNS Software* E. Gavron

1536 *Common DNS Implementation Errors and Suggested Fixes* A. Kumar, J. Postel, C. Neuman, P. Danzig, S. Miller

1537 *Common DNS Data File Configuration Errors* P. Beertema

1540 *Internet Official Protocol Standards* J. Postel

1541 *Dynamic Host Configuration Protocol* R. Droms

1542 *Clarifications and Extensions for the Bootstrap Protocol* W. Wimer

1571 *Telnet Environment Option Interoperability Issues* D. Borman

1572 *Telnet Environment Option* S. Alexander

1573 *Evolution of the Interfaces Group of MIB-II* K. McCloghrie, F. Kastenholz

1577 *Classical IP and ARP over ATM* M. Laubach

1583 *OSPF Version 2* J. Moy

1591 *Domain Name System Structure and Delegation* J. Postel

1592 *Simple Network Management Protocol Distributed Protocol Interface Version 2.0* B. Wijnen, G. Carpenter, K. Curran, A. Sehgal, G. Waters

1594 *FYI on Questions and Answers— Answers to Commonly Asked "New Internet User" Questions* A. Marine, J. Reynolds, G. Malkin

1644 *T/TCP — TCP Extensions for Transactions Functional Specification* R. Braden

1646 *TN3270 Extensions for LUname and Printer Selection* C. Graves, T. Butts, M. Angel

1647 *TN3270 Enhancements* B. Kelly

1652 *SMTP Service Extension for 8bit-MIMEtransport* J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker

1664 *Using the Internet DNS to Distribute RFC1327 Mail Address Mapping Tables* C. Allochio, A. Bonito, B. Cole, S. Giordano, R. Hagens

1693 *An Extension to TCP: Partial Order Service* T. Connolly, P. Amer, P. Conrad

1695 *Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2* M. Ahmed, K. Tesink

1701 *Generic Routing Encapsulation (GRE)* S. Hanks, T. Li, D. Farinacci, P. Traina

1702 *Generic Routing Encapsulation over IPv4 networks* S. Hanks, T. Li, D. Farinacci, P. Traina

1706 *DNS NSAP Resource Records* B. Manning, R. Colella

1712 *DNS Encoding of Geographical Location* C. Farrell, M. Schulze, S. Pleitner D. Baldoni

1713 *Tools for DNS debugging* A. Romao

1723 *RIP Version 2—Carrying Additional Information* G. Malkin

1752 *The Recommendation for the IP Next Generation Protocol* S. Bradner, A. Mankin

1766 *Tags for the Identification of Languages* H. Alvestrand

1771 *A Border Gateway Protocol 4 (BGP-4)* Y. Rekhter, T. Li

1794 *DNS Support for Load Balancing* T. Brisco

1819 *Internet Stream Protocol Version 2 (ST2) Protocol Specification—Version ST2+* L. Delgrossi, L. Berger Eds.

1826 *IP Authentication Header* R. Atkinson

1828 *IP Authentication using Keyed MD5* P. Metzger, W. Simpson

1829 *The ESP DES-CBC Transform* P. Karn, P. Metzger, W. Simpson

1830 *SMTP Service Extensions for Transmission of Large and Binary MIME Messages* G. Vaudreuil

1832 *XDR: External Data Representation Standard* R. Srinivasan

1850 *OSPF Version 2 Management Information Base* F. Baker, R. Coltun

1854 *SMTP Service Extension for Command Pipelining* N. Freed

1869 *SMTP Service Extensions* J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker

1870 *SMTP Service Extension for Message Size Declaration* J. Klensin, N. Freed, K. Moore

1876 *A Means for Expressing Location Information in the Domain Name System* C. Davis, P. Vixie, T. Goodwin, I. Dickinson

1883 *Internet Protocol, Version 6 (IPv6) Specification* S. Deering, R. Hinden

1884 *IP Version 6 Addressing Architecture* R. Hinden, S. Deering, Eds.

1886 *DNS Extensions to support IP version 6* S. Thomson, C. Huitema

1888 *OSI NSAPs and IPv6* J. Bound, B. Carpenter, D. Harrington, J. Houldsworth, A. Lloyd

1891 *SMTP Service Extension for Delivery Status Notifications* K. Moore

1892 *The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages* G. Vaudreuil

1894 *An Extensible Message Format for Delivery Status Notifications*K. Moore, G. Vaudreuil

1901 *Introduction to Community-based SNMPv2* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1902 *Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1903 *Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1904 *Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1905 *Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1906 *Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1907 *Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1908 *Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1912 *Common DNS Operational and Configuration Errors* D. Barr

1918 *Address Allocation for Private Internets* Y. Rekhter, B. Moskowitz, D. Karrenberg, G.J. de Groot, E. Lear

1928 *SOCKS Protocol Version 5* M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones

1930 *Guidelines for creation, selection, and registration of an Autonomous System (AS)* J. Hawkinson, T. Bates

1939 *Post Office Protocol-Version 3* J. Myers, M. Rose

1981 *Path MTU Discovery for IP version 6* J. McCann, S. Deering, J. Mogul

1982 *Serial Number Arithmetic* R. Elz, R. Bush

1985 *SMTP Service Extension for Remote Message Queue Starting* J. De Winter

1995 *Incremental Zone Transfer in DNS* M. Ohta

1996 *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)* P. Vixie

2010 *Operational Criteria for Root Name Servers* B. Manning, P. Vixie

2011 *SNMPv2 Management Information Base for the Internet Protocol using SMIv2* K. McCloghrie, Ed.

2012 *SNMPv2 Management Information Base for the Transmission Control Protocol using SMIv2* K. McCloghrie, Ed.

2013 *SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2* K. McCloghrie, Ed.

2018 *TCP Selective Acknowledgement Options* M. Mathis, J. Mahdavi, S. Floyd, A. Romanow

2026 *The Internet Standards Process — Revision 3* S. Bradner

2030 *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI* D. Mills

2033 *Local Mail Transfer Protocol* J. Myers

2034 *SMTP Service Extension for Returning Enhanced Error Codes*N. Freed

2040 *The RC5, RC5–CBC, RC-5–CBC-Pad, and RC5–CTS Algorithms*R. Baldwin, R. Rivest

2045 *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies* N. Freed, N. Borenstein

2052 *A DNS RR for specifying the location of services (DNS SRV)* A. Gulbrandsen, P. Vixie

2065 *Domain Name System Security Extensions* D. Eastlake 3rd, C. Kaufman

2066 *TELNET CHARSET Option* R. Gellens

2080 *RIPng for IPv6* G. Malkin, R. Minnear

2096 *IP Forwarding Table MIB* F. Baker

2104 *HMAC: Keyed-Hashing for Message Authentication* H. Krawczyk, M. Bellare, R. Canetti

2119 *Keywords for use in RFCs to Indicate Requirement Levels* S. Bradner

2132 *DHCP Options and BOOTP Vendor Extensions* S. Alexander, R. Droms

2133 *Basic Socket Interface Extensions for IPv6* R. Gilligan, S. Thomson, J. Bound, W. Stevens

2136 *Dynamic Updates in the Domain Name System (DNS UPDATE)* P. Vixie, Ed., S. Thomson, Y. Rekhter, J. Bound

2137 *Secure Domain Name System Dynamic Update* D. Eastlake 3rd

2163 *Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM)* C. Allocchio

2168 *Resolution of Uniform Resource Identifiers using the Domain Name System* R. Daniel, M. Mealling

2178 *OSPF Version 2* J. Moy

2181 *Clarifications to the DNS Specification* R. Elz, R. Bush

2205 *Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification* R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin

2210 *The Use of RSVP with IETF Integrated Services* J. Wroclawski

2211 *Specification of the Controlled-Load Network Element Service* J. Wroclawski

2212 *Specification of Guaranteed Quality of Service* S. Shenker, C. Partridge, R. Guerin

2215 *General Characterization Parameters for Integrated Service Network Elements* S. Shenker, J. Wroclawski

2217 *Telnet Com Port Control Option* G. Clarke

2219 *Use of DNS Aliases for Network Services* M. Hamilton, R. Wright

2228 *FTP Security Extensions* M. Horowitz, S. Lunt

2230 *Key Exchange Delegation Record for the DNS* R. Atkinson

2233 *The Interfaces Group MIB using SMIv2* K. McCloghrie, F. Kastenholz

2240 *A Legal Basis for Domain Name Allocation* O. Vaughn

2246 *The TLS Protocol Version 1.0* T. Dierks, C. Allen

2251 *Lightweight Directory Access Protocol (v3)* M. Wahl, T. Howes, S. Kille

2253 *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names* M. Wahl, S. Kille, T. Howes

2254 *The String Representation of LDAP Search Filters* T. Howes

2261 *An Architecture for Describing SNMP Management Frameworks* D. Harrington, R. Presuhn, B. Wijnen

2262 *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)* J. Case, D. Harrington, R. Presuhn, B. Wijnen

2271 *An Architecture for Describing SNMP Management Frameworks* D. Harrington, R. Presuhn, B. Wijnen

2273 *SNMPv3 Applications* D. Levi, P. Meyer, B. Stewartz

2274 *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)* U. Blumenthal, B. Wijnen

2275 *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)* B. Wijnen, R. Presuhn, K. McCloghrie

2292 *Advanced Sockets API for IPv6* W. Stevens, M. Thomas

2308 *Negative Caching of DNS Queries (DNS NCACHE)* M. Andrews

2317 *Classless IN-ADDR.ARPA delegation* H. Eidnes, G. de Groot, P. Vixie

2320 *Definitions of Managed Objects for Classical IP and ARP Over ATM Using SMIv2 (IPOA-MIB)* M. Greene, J. Luciani, K. White, T. Kuo

2328 *OSPF Version 2* J. Moy

2345 *Domain Names and Company Name Retrieval* J. Klensin, T. Wolf, G. Oglesby

2352 *A Convention for Using Legal Names as Domain Names* O. Vaughn

2355 *TN3270 Enhancements* B. Kelly

2358 *Definitions of Managed Objects for the Ethernet-like Interface Types* J. Flick, J. Johnson

2373 *IP Version 6 Addressing Architecture* R. Hinden, S. Deering

2374 *An IPv6 Aggregatable Global Unicast Address Format* R. Hinden, M. O'Dell, S. Deering

2375 *IPv6 Multicast Address Assignments* R. Hinden, S. Deering

2385 *Protection of BGP Sessions via the TCP MD5 Signature Option*A. Hefferman

2389 *Feature negotiation mechanism for the File Transfer Protocol* P. Hethmon, R. Elz

2401 *Security Architecture for Internet Protocol* S. Kent, R. Atkinson

2402 *IP Authentication Header* S. Kent, R. Atkinson

2403 *The Use of HMAC-MD5–96 within ESP and AH* C. Madson, R. Glenn

2404 *The Use of HMAC-SHA–1–96 within ESP and AH* C. Madson, R. Glenn

2405 *The ESP DES-CBC Cipher Algorithm With Explicit IV* C. Madson, N. Doraswamy

2406 *IP Encapsulating Security Payload (ESP)* S. Kent, R. Atkinson

2407 *The Internet IP Security Domain of Interpretation for ISAKMP*D. Piper

2408 *Internet Security Association and Key Management Protocol (ISAKMP)* D. Maughan, M. Schertler, M. Schneider, J. Turner

2409 *The Internet Key Exchange (IKE)* D. Harkins, D. Carrel

2410 *The NULL Encryption Algorithm and Its Use With IPsec* R. Glenn, S. Kent,

2428 *FTP Extensions for IPv6 and NATs* M. Allman, S. Ostermann, C. Metz

2445 *Internet Calendaring and Scheduling Core Object Specification (iCalendar)* F. Dawson, D. Stenerson

2459 *Internet X.509 Public Key Infrastructure Certificate and CRL Profile* R. Housley, W. Ford, W. Polk, D. Solo

2460 *Internet Protocol, Version 6 (IPv6) Specification* S. Deering, R. Hinden

2461 *Neighbor Discovery for IP Version 6 (IPv6)* T. Narten, E. Nordmark, W. Simpson

2462 *IPv6 Stateless Address Autoconfiguration* S. Thomson, T. Narten

2463 *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification* A. Conta, S. Deering

2464 *Transmission of IPv6 Packets over Ethernet Networks* M. Crawford

2466 *Management Information Base for IP Version 6: ICMPv6 Group* D. Haskin, S. Onishi

2476 *Message Submission* R. Gellens, J. Klensin

2487 *SMTP Service Extension for Secure SMTP over TLS* P. Hoffman

2505 *Anti-Spam Recommendations for SMTP MTAs* G. Lindberg

2523 *Photuris: Extended Schemes and Attributes* P. Karn, W. Simpson

2535 *Domain Name System Security Extensions* D. Eastlake 3rd

2538 *Storing Certificates in the Domain Name System (DNS)* D. Eastlake 3rd, O. Gudmundsson

2539 *Storage of Diffie-Hellman Keys in the Domain Name System (DNS)* D. Eastlake 3rd

2540 *Detached Domain Name System (DNS) Information* D. Eastlake 3rd

2554 *SMTP Service Extension for Authentication* J. Myers

2570 *Introduction to Version 3 of the Internet-standard Network Management Framework* J. Case, R. Mundy, D. Partain, B. Stewart

2571 *An Architecture for Describing SNMP Management Frameworks* B. Wijnen, D. Harrington, R. Presuhn

2572 *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)* J. Case, D. Harrington, R. Presuhn, B. Wijnen

2573 *SNMP Applications* D. Levi, P. Meyer, B. Stewart

2574 *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)* U. Blumenthal, B. Wijnen

2575 *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)* B. Wijnen, R. Presuhn, K. McCloghrie

2576 *Co-Existence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework* R. Frye, D. Levi, S. Routhier, B. Wijnen

2578 *Structure of Management Information Version 2 (SMIv2)* K. McCloghrie, D. Perkins, J. Schoenwaelder

2579 *Textual Conventions for SMIv2* K. McCloghrie, D. Perkins, J. Schoenwaelder

2580 *Conformance Statements for SMIv2* K. McCloghrie, D. Perkins, J. Schoenwaelder

2581 *TCP Congestion Control* M. Allman, V. Paxson, W. Stevens

2583 *Guidelines for Next Hop Client (NHC) Developers* R. Carlson, L. Winkler

2591 *Definitions of Managed Objects for Scheduling Management Operations* D. Levi, J. Schoenwaelder

2625 *IP and ARP over Fibre Channel* M. Rajagopal, R. Bhagwat, W. Rickard

2635 *Don't SPEW A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam*)* S. Hambridge, A. Lunde

2637 *Point-to-Point Tunneling Protocol* K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, G. Zorn

2640 *Internationalization of the File Transfer Protocol* B. Curtin

2665 *Definitions of Managed Objects for the Ethernet-like Interface Types* J. Flick, J. Johnson

2671 *Extension Mechanisms for DNS (EDNS0)* P. Vixie

2672 *Non-Terminal DNS Name Redirection* M. Crawford

2675 *IPv6 Jumbograms* D. Borman, S. Deering, R. Hinden

2710 *Multicast Listener Discovery (MLD) for IPv6* S. Deering, W. Fenner, B. Haberman

2711 *IPv6 Router Alert Option* C. Partridge, A. Jackson

2740 *OSPF for IPv6* R. Coltun, D. Ferguson, J. Moy

2753 *A Framework for Policy-based Admission Control* R. Yavatkar, D. Pendarakis, R. Guerin

2758 *Definitions of Managed Objects for Service Level Agreements Performance Monitoring* K. White

2782 *A DNS RR for specifying the location of services (DNS SRV)* A. Gubrandsen, P. Vixix, L. Esibov

2821 *Simple Mail Transfer Protocol* J. Klensin, Ed.

2822 *Internet Message Format* P. Resnick, Ed.

2840 *TELNET KERMIT OPTION* J. Altman, F. da Cruz

2845 *Secret Key Transaction Authentication for DNS (TSIG)* P. Vixie, O. Gudmundsson, D. Eastlake 3rd, B. Wellington

2851 *Textual Conventions for Internet Network Addresses* M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder

2852 *Deliver By SMTP Service Extension* D. Newman

2874 *DNS Extensions to Support IPv6 Address Aggregation and Renumbering* M. Crawford, C. Huitema

2915 *The Naming Authority Pointer (NAPTR) DNS Resource Record* M. Mealling, R. Daniel

2920 *SMTP Service Extension for Command Pipelining* N. Freed

2930 *Secret Key Establishment for DNS (TKEY RR)* D. Eastlake, 3rd

2941 *Telnet Authentication Option* T. Ts'o, ed., J. Altman

2942 *Telnet Authentication: Kerberos Version 5* T. Ts'o

2946 *Telnet Data Encryption Option* T. Ts'o

2952 *Telnet Encryption: DES 64 bit Cipher Feedback* T. Ts'o

2953 *Telnet Encryption: DES 64 bit Output Feedback* T. Ts'o

2992 *Analysis of an Equal-Cost Multi-Path Algorithm* C. Hopps

3019 *IP Version 6 Management Information Base for The Multicast Listener Discovery Protocol* B. Haberman, R. Worzella

3060 *Policy Core Information Model—Version 1 Specification* B. Moore, E. Ellesson, J. Strassner, A. Westerinen

3152 *Delegation of IP6.ARPA* R. Bush

3291 *Textual Conventions for Internet Network Addresses* M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder

3363 *Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System* R. Bush, A. Durand, B. Fink, O. Gudmundsson, T. Hain

3390 *Increasing TCP's Initial Window* M. Allman, S. Floyd, C. Partridge

3411 *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks* D. Harrington, R. Presuhn, B. Wijnen

3412 *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)* J. Case, D. Harrington, R. Presuhn, B. Wijnen

3413 *Simple Network Management Protocol (SNMP) Applications* D. Levi, P. Meyer, B. Stewart

3414 *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)* U. Blumenthal, B. Wijnen

3415 *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)* B. Wijnen, R. Presuhn, K. McCloghrie

3419 *Textual Conventions for Transport Addresses* M. Daniele, J. Schoenwaelder

3484 *Default Address Selection for Internet Protocol version 6 (IPv6)* R. Draves

3493 *Basic Socket Interface Extensions for IPv6* R. Gilligan, S. Thomson, J. Bound, J. McCann, W. Stevens

3513 *Internet Protocol Version 6 (IPv6) Addressing Architecture* R. Hinden, S. Deering

3542 *Advanced Sockets Application Programming Interface (API) for IPv6* W. Richard Stevens, M. Thomas, E. Nordmark, T. Jinmei

3658 *Delegation Signer (DS) Resource Record (RR)* O. Gudmundsson

3715 *IPsec-Network Address Translation (NAT) Compatibility Requirements* B. Aboba, W. Dixon

3947 *Negotiation of NAT-Traversal in the IKE* T. Kivinen, B. Swander, A. Huttunen, V. Volpe

3948 *UDP Encapsulation of IPsec ESP Packets* A. Huttunen, B. Swander, V. Volpe, L. DiBurro, M. Stenberg

Internet drafts

Internet drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Other groups may also distribute working documents as Internet drafts. You can see Internet drafts at http://www.ietf.org/ID.html.

Several areas of IPv6 implementation include elements of the following Internet drafts and are subject to change during the RFC review process.

Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification
A. Conta, S. Deering

Appendix J. Information APARs

This appendix lists information APARs for IP and SNA documents.

Notes:

1. Information APARs contain updates to previous editions of the manuals listed below. Documents updated for V1R7 are complete except for the updates contained in the information APARs that might be issued after V1R7 documents went to press.
2. Information APARs are predefined for z/OS V1R7 Communications Server and might not contain updates.
3. Information APARs for z/OS documents are in the document called *z/OS and z/OS.e DOC APAR and PTF ++HOLD Documentation*, which can be found at http://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS.

Information APARs for IP documents

Table 43 lists information APARs for IP documents. For information APARs for V1R7, see http://www.ibm.com/support/docview.wss?uid=swg21178966.

Table 43. IP information APARs for z/OS Communications Server

Title	V1R6	V1R5	V1R4
New Function Summary (both IP and SNA)	II13824		
Quick Reference (both IP and SNA)	II13831		II13246
IP and SNA Codes	II13842		II13254
IP API Guide	II13844	II13577	II13255 II13790
IP CICS® Sockets Guide		II13578	II13257
IP Configuration Guide	II13826	II13568	II13244 II13541 II13652 II13646
IP Configuration Reference	II13827	II13569 II13789	II13245 II13521 II13647 II13739
IP Diagnosis	II13836	II13571	II13249 II13493
IP Messages Volume 1	II13838	II13572	II13624 II13250
IP Messages Volume 2	II13839	II13573	II13251
IP Messages Volume 3	II13840	II13574	II13252
IP Messages Volume 4	II13841	II13575	II13253 II13628
IP Migration		II13566	II13242 II13738
IP Network and Application Design Guide	II13825	II13567	II13243

Table 43. IP information APARs for z/OS Communications Server (continued)

Title	V1R6	V1R5	V1R4
IP Network Print Facility			
IP Programmer's Reference	II13843	II13581	II13256
IP User's Guide and Commands	II13832	II13570	II13247
IP System Admin Commands	II13833	II13580	II13248 II13792

Information APARs for SNA documents

Table 44 lists information APARs for SNA documents. For information APARs for V1R7, see http://www.ibm.com/support/docview.wss?uid=swg21178966.

Table 44. SNA information APARs for z/OS Communications Server

Title	V1R6	V1R5	V1R4
New Function Summary (both IP and SNA)	II13824		
Quick Reference (both IP and SNA)	II13831		II13246
IP and SNA Codes	II13842		II13254
SNA Customization	II13857	II13560	II13240
SNA Diagnosis		II13558	II13236 II13735
SNA Diagnosis, Vol. 1: Techniques and Procedures	II13852		
SNA Diagnosis, Vol. 2: FFST Dumps and the VIT	II13853		
SNA Messages	II13854	II13559	II13238 II13736
SNA Network Implementation Guide	II13849	II13555	II13234 II13733
SNA Operation	II13851	II13557	II13237
SNA Migration		II13554	II13233 II13732
SNA Programming	II13858		II13241
SNA Resource Definition Reference	II13850	II13556	II13235 II13734
SNA Data Areas, Vol. 1 and 2			II13239
SNA Data Areas, 1	II13855		
SNA Data Areas, 2	II13856		

Other information APARs

Table 45 lists information APARs not related to documents.

Table 45. Non-document information APARs

Content	Number
Index to APARs that list recommended VTAM maintenance	II11220

Table 45. Non-document information APARs (continued)

Content	Number
Index to APARs that list trace and dump requests for VTAM problems	II13202
Index of Communication Server IP information APARs	II12028
MPC and CTC	II01501
Collecting TCPIP CTRACEs	II12014
CSM for VTAM	II13442
CSM for TCP/IP	II13951
DLUR/DLUS for z/OS V1R2, V1R4, and V1R5	II12986, II13456, and II13783
DOCUMENTATION REQUIRED FOR OSA/2, OSA EXPRESS AND OSA QDIO	II13016
DYNAMIC VIPA (BIND)	II13215
DNS — common problems and solutions	II13453
Enterprise Extender	II12223
FTPing doc to z/OS Support	II12030
FTP problems	II12079
Generic resources	II10986
HPR	II10953
iQDIO	II13142
LPR problems	II12022
MNPS	II10370
NCPROUTE problems	II12025
OMPROUTE	II12026
PASCAL API	II11814
Performance	II11710 II11711 II11712
Resolver	II13398 II13399 II13452
Socket API	II11996 II12020
SMTP problems	II12023
SNMP	II13477 II13478
SYSLOGD howto	II12021
TCPIP connection states	II12449
Telnet	II11574 II13135
TN3270 TELNET SSL common problems	II13369

Appendix K. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

`www.ibm.com/servers/eserver/zseries/zos/bkserv/`

Notices

IBM may not offer all of the products, services, or features discussed in this document. 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 to 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 information 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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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:

Site Counsel
IBM Corporation
P.O. Box 12195
3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195
U.S.A

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 information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application

programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

IBM is required to include the following statements in order to distribute portions of this document and the software described herein to which contributions have been made by The University of California. Portions herein © Copyright 1979, 1980, 1983, 1986, Regents of the University of California. Reproduced by permission. Portions herein were developed at the Electrical Engineering and Computer Sciences Department at the Berkeley campus of the University of California under the auspices of the Regents of the University of California.

Portions of this publication relating to RPC are Copyright © Sun Microsystems, Inc., 1988, 1989.

Some portions of this publication relating to X Window System** are Copyright © 1987, 1988 by Digital Equipment Corporation, Maynard, Massachusetts, and the Massachusetts Institute Of Technology, Cambridge, Massachusetts. All Rights Reserved.

Some portions of this publication relating to X Window System are Copyright © 1986, 1987, 1988 by Hewlett-Packard Corporation.

Permission to use, copy, modify, and distribute the M.I.T., Digital Equipment Corporation, and Hewlett-Packard Corporation portions of this software and its documentation for any purpose without fee is hereby granted, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the names of M.I.T., Digital, and Hewlett-Packard not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T., Digital, and Hewlett-Packard make no representation about the suitability of this software for any purpose. It is provided ″as is″ without express or implied warranty.

Copyright © 1983, 1995-1997 Eric P. Allman

Copyright © 1988, 1993 The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   ```
   This product includes software developed by the University of
   California, Berkeley and its contributors.
   ```

4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software program contains code, and/or derivatives or modifications of code originating from the software program "Popper." Popper is Copyright ©1989-1991 The Regents of the University of California, All Rights Reserved. Popper was created by Austin Shelton, Information Systems and Technology, University of California, Berkeley.

Permission from the Regents of the University of California to use, copy, modify, and distribute the "Popper" software contained herein for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies. HOWEVER, ADDITIONAL PERMISSIONS MAY BE NECESSARY FROM OTHER PERSONS OR ENTITIES, TO USE DERIVATIVES OR MODIFICATIONS OF POPPER.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THE POPPER SOFTWARE, OR ITS DERIVATIVES OR MODIFICATIONS, AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE POPPER SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by the University of California, Berkeley. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   `This product includes software developed by the University of California, Berkeley and its contributors.`
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Furthermore

if you modify this software you must label your software as modified software and not distribute it in such a fashion that it might be confused with the original M.I.T. software. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided ″as is″ without express or implied warranty.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of FundsXpress not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. FundsXpress makes no representations about the suitability of this software for any purpose. It is provided ″as is″ without express or implied warranty.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED ″AS IS″ AND INTERNET SOFTWARE CONSORTIUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL INTERNET SOFTWARE CONSORTIUM BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscape's SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be

given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: ″This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)″. The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related.
4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include acknowledgement:

   ″This product includes software written by Tim Hudson (tjh@cryptsoft.com)″

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The license and distribution terms for any publicly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution license [including the GNU Public License.]

This product includes cryptographic software written by Eric Young.

Copyright © 1999, 2000 Internet Software Consortium.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED ″AS IS″ AND INTERNET SOFTWARE CONSORTIUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL INTERNET SOFTWARE CONSORTIUM BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

If you are viewing this information softcopy, photographs and color illustrations may not appear.

You can obtain softcopy from the z/OS Collection (SK3T-4269), which contains BookManager and PDF formats of unlicensed books and the z/OS Licensed Product Library (LK3T-4307), which contains BookManager and PDF formats of licensed books.

Programming interface information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/OS Communications Server.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

Advanced Peer-to-Peer Networking
AFP
AD/Cycle
AIX
AIX/ESA
AnyNet
APL2
AS/400
AT
BookManager
BookMaster
C/370
CICS
CICS/ESA
C/MVS
Common User Access
C Set ++
CT
CUA
DB2
DFSMSdfp
DFSMShsm
DFSMS/MVS
DPI
Domino
DRDA
Enterprise Systems Architecture/370
ESCON
eServer
ES/3090
ES/9000
ES/9370
EtherStreamer
Extended Services
FFST
FFST/2
First Failure Support Technology
GDDM
IBM
IBMLink
IMS
IMS/ESA
HiperSockets
Language Environment
LANStreamer
Library Reader
LPDA
Micro Channel
Multiprise
MVS
MVS/DFP
MVS/ESA
MVS/SP
MVS/XA
NetView
Network Station
Nways
Notes
OfficeVision/MVS
OfficeVision/VM
Open Class
OS/2
OS/390
OS/400
Parallel Sysplex
PR/SM
PROFS
PS/2
RACF
Redbooks
Resource Link
RETAIN
RISC System/6000
RMF
RS/6000
S/370
S/390
S/390 Parallel Enterprise Server
SAA
SecureWay
SP
SP2
SQL/DS
System/360
System/370
System/390
SystemView
Tivoli
TURBOWAYS
VM/ESA
VSE/ESA
VTAM
WebSphere
XT
z/Architecture
z/OS
zSeries
z/VM
400
3090
3890

DB2 and NetView are registered trademarks of International Business Machines Corporation or Tivoli Systems Inc. in the U.S., other countries, or both.

The following terms are trademarks of other companies:

ATM is a trademark of Adobe Systems, Incorporated.

BSC is a trademark of BusiSoft Corporation.

CSA is a trademark of Canadian Standards Association.

DCE is a trademark of The Open Software Foundation.

HYPERchannel is a trademark of Network Systems Corporation.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

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

Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

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

Bibliography

z/OS Communications Server information

This section contains descriptions of the documents in the z/OS Communications Server library.

z/OS Communications Server documentation is available:

- Online at the z/OS Internet Library web page at http://www.ibm.com/servers/eserver/zseries/zos/bkserv
- In softcopy on CD-ROM collections. See "Softcopy information" on page xviii.

z/OS Communications Server library

z/OS Communications Server documents are available on the CD-ROM accompanying z/OS (SK3T-4269 or SK3T-4307). Unlicensed documents can be viewed at the z/OS Internet library site.

Updates to documents are available on RETAIN® and in information APARs (info APARs). See Appendix J, "Information APARs," on page 369 for a list of the documents and the info APARs associated with them.

Info APARs for z/OS documents are in the document called *z/OS and z/OS.e DOC APAR and PTF ++HOLD Documentation* which can be found at http://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/ BOOKS/ZIDOCMST/CCONTENTS.

Planning

Title	Number	Description
z/OS Communications Server: New Function Summary	GC31-8771	This document is intended to help you plan for new IP for SNA function, whether you are migrating from a previous version or installing z/OS for the first time. It summarizes what is new in the release and identifies the suggested and required modifications needed to use the enhanced functions.
z/OS Communications Server: IPv6 Network and Application Design Guide	SC31-8885	This document is a high-level introduction to IPv6. It describes concepts of z/OS Communications Server's support of IPv6, coexistence with IPv4, and migration issues.

Resource definition, configuration, and tuning

Title	Number	Description
z/OS Communications Server: IP Configuration Guide	SC31-8775	This document describes the major concepts involved in understanding and configuring an IP network. Familiarity with the z/OS operating system, IP protocols, z/OS UNIX System Services, and IBM Time Sharing Option (TSO) is recommended. Use this document in conjunction with the *z/OS Communications Server: IP Configuration Reference*.

Title	Number	Description
z/OS Communications Server: IP Configuration Reference	SC31-8776	This document presents information for people who want to administer and maintain IP. Use this document in conjunction with the *z/OS Communications Server: IP Configuration Guide*. The information in this document includes: • TCP/IP configuration data sets • Configuration statements • Translation tables • SMF records • Protocol number and port assignments
z/OS Communications Server: SNA Network Implementation Guide	SC31-8777	This document presents the major concepts involved in implementing an SNA network. Use this document in conjunction with the *z/OS Communications Server: SNA Resource Definition Reference*.
z/OS Communications Server: SNA Resource Definition Reference	SC31-8778	This document describes each SNA definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect SNA. Use this document in conjunction with the *z/OS Communications Server: SNA Network Implementation Guide*.
z/OS Communications Server: SNA Resource Definition Samples	SC31-8836	This document contains sample definitions to help you implement SNA functions in your networks, and includes sample major node definitions.
z/OS Communications Server: AnyNet SNA over TCP/IP	SC31-8832	This guide provides information to help you install, configure, use, and diagnose SNA over TCP/IP.
z/OS Communications Server: AnyNet Sockets over SNA	SC31-8831	This guide provides information to help you install, configure, use, and diagnose sockets over SNA. It also provides information to help you prepare application programs to use sockets over SNA.
z/OS Communications Server: IP Network Print Facility	SC31-8833	This document is for system programmers and network administrators who need to prepare their network to route SNA, JES2, or JES3 printer output to remote printers using TCP/IP Services.

Operation

Title	Number	Description
z/OS Communications Server: IP User's Guide and Commands	SC31-8780	This document describes how to use TCP/IP applications. It contains requests that allow a user to log on to a remote host using Telnet, transfer data sets using FTP, send and receive electronic mail, print on remote printers, and authenticate network users.
z/OS Communications Server: IP System Administrator's Commands	SC31-8781	This document describes the functions and commands helpful in configuring or monitoring your system. It contains system administrator's commands, such as TSO NETSTAT, PING, TRACERTE and their UNIX counterparts. It also includes TSO and MVS commands commonly used during the IP configuration process.
z/OS Communications Server: SNA Operation	SC31-8779	This document serves as a reference for programmers and operators requiring detailed information about specific operator commands.
z/OS Communications Server: Quick Reference	SX75-0124	This document contains essential information about SNA and IP commands.

Customization

Title	Number	Description
z/OS Communications Server: SNA Customization	SC31-6854	This document enables you to customize SNA, and includes the following: • Communication network management (CNM) routing table • Logon-interpret routine requirements • Logon manager installation-wide exit routine for the CLU search exit • TSO/SNA installation-wide exit routines • SNA installation-wide exit routines

Writing application programs

Title	Number	Description
z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference	SC31-8788	This document describes the syntax and semantics of program source code necessary to write your own application programming interface (API) into TCP/IP. You can use this interface as the communication base for writing your own client or server application. You can also use this document to adapt your existing applications to communicate with each other using sockets over TCP/IP.
z/OS Communications Server: IP CICS Sockets Guide	SC31-8807	This document is for programmers who want to set up, write application programs for, and diagnose problems with the socket interface for CICS using z/OS TCP/IP.
z/OS Communications Server: IP IMS Sockets Guide	SC31-8830	This document is for programmers who want application programs that use the IMS™ TCP/IP application development services provided by IBM's TCP/IP Services.
z/OS Communications Server: IP Programmer's Guide and Reference	SC31-8787	This document describes the syntax and semantics of a set of high-level application functions that you can use to program your own applications in a TCP/IP environment. These functions provide support for application facilities, such as user authentication, distributed databases, distributed processing, network management, and device sharing. Familiarity with the z/OS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.
z/OS Communications Server: SNA Programming	SC31-8829	This document describes how to use SNA macroinstructions to send data to and receive data from (1) a terminal in either the same or a different domain, or (2) another application program in either the same or a different domain.
z/OS Communications Server: SNA Programmer's LU 6.2 Guide	SC31-8811	This document describes how to use the SNA LU 6.2 application programming interface for host application programs. This document applies to programs that use only LU 6.2 sessions or that use LU 6.2 sessions along with other session types. (Only LU 6.2 sessions are covered in this document.)
z/OS Communications Server: SNA Programmer's LU 6.2 Reference	SC31-8810	This document provides reference material for the SNA LU 6.2 programming interface for host application programs.
z/OS Communications Server: CSM Guide	SC31-8808	This document describes how applications use the communications storage manager.

Title	Number	Description
z/OS Communications Server: CMIP Services and Topology Agent Guide	SC31-8828	This document describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The document provides guide and reference information about CMIP services and the SNA topology agent.

Diagnosis

Title	Number	Description
z/OS Communications Server: IP Diagnosis Guide	GC31-8782	This document explains how to diagnose TCP/IP problems and how to determine whether a specific problem is in the TCP/IP product code. It explains how to gather information for and describe problems to the IBM Software Support Center.
z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures and *z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT*	GC31-6850 GC31-6851	These documents help you identify an SNA problem, classify it, and collect information about it before you call the IBM Support Center. The information collected includes traces, dumps, and other problem documentation.
z/OS Communications Server: SNA Data Areas Volume 1 and *z/OS Communications Server: SNA Data Areas Volume 2*	GC31-6852 GC31-6853	These documents describe SNA data areas and can be used to read an SNA dump. They are intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.

Messages and codes

Title	Number	Description
z/OS Communications Server: SNA Messages	SC31-8790	This document describes the ELM, IKT, IST, ISU, IUT, IVT, and USS messages. Other information in this document includes: • Command and RU types in SNA messages • Node and ID types in SNA messages • Supplemental message-related information
z/OS Communications Server: IP Messages Volume 1 (EZA)	SC31-8783	This volume contains TCP/IP messages beginning with EZA.
z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)	SC31-8784	This volume contains TCP/IP messages beginning with EZB or EZD.
z/OS Communications Server: IP Messages Volume 3 (EZY)	SC31-8785	This volume contains TCP/IP messages beginning with EZY.
z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)	SC31-8786	This volume contains TCP/IP messages beginning with EZZ and SNM.
z/OS Communications Server: IP and SNA Codes	SC31-8791	This document describes codes and other information that appear in z/OS Communications Server messages.

APPC Application Suite

Title	Number	Description
z/OS Communications Server: APPC Application Suite User's Guide	SC31-8809	This documents the end-user interface (concepts, commands, and messages) for the AFTP, ANAME, and APING facilities of the APPC application suite. Although its primary audience is the end user, administrators and application programmers may also find it useful.

Title	Number	Description
z/OS Communications Server: APPC Application Suite Administration	SC31-8835	This document contains the information that administrators need to configure the APPC application suite and to manage the APING, ANAME, AFTP, and A3270 servers.
z/OS Communications Server: APPC Application Suite Programming	SC31-8834	This document provides the information application programmers need to add the functions of the AFTP and ANAME APIs to their application programs.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

Z

Communicating Your Comments to IBM

If you especially like or dislike anything about this document, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. However, the comments you send should pertain to only the information in this manual 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.

Please send your comments to us in either of the following ways:

- If you prefer to send comments by FAX, use this number: 1+919-254-4028
- If you prefer to send comments electronically, use this address:
  - comsvrcf@us.ibm.com.
- If you prefer to send comments by post, use this address:

```
International Business Machines Corporation
Attn: z/OS Communications Server Information Development
P.O. Box 12195, 3039 Cornwallis Road
Department AKCA, Building 501
Research Triangle Park, North Carolina 27709-2195
```

Make sure to include the following in your note:

- Title and publication number of this document
- Page number or topic to which your comment applies.

Program Number: 5694–A01 and 5655–G52

Printed in USA

SC31-8828-03

Spine information:

z/OS Communications Server

z/OS V1R7.0 Comm Svr: CMIP Services and Topology Agent Guide

Version 1
Release 7