AS/400

Simple Network Management Protocol (SNMP) Support

Version 4

SC41-5412-00

AS/400

Simple Network Management Protocol (SNMP) Support

Version 4

SC41-5412-00

Take Note!

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

First Edition (August 1997)

This edition applies to the licensed program Operating System/400, (Program 5769-SS1), Version 4 Release 1 Modification 1, and to all subsequent releases and modifications until otherwise indicated in new editions.

Make sure that you are using the proper edition for the level of the product.

Order publications through your IBM representative or the IBM branch serving your locality. If you live in the United States, Puerto Rico, or Guam, you can order publications through the IBM Software Manufacturing Solutions at 800+879-2755. Publications are not stocked at the address given below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication. You can also mail your comments to the following address:

IBM Corporation
Attention Department 542
IDCLERK
3605 Highway 52 N
Rochester, MN 55901-7829 USA

or you can fax your comments to:

United States and Canada: 800+937-3430
Other countries: (+1)+507+253-5192

If you have access to Internet, you can send your comments electronically to IDCLERK@RCHVMW2.VNET.IBM.COM; IBMMAIL, to IBMMAIL(USIB56RZ).

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

Part 1. Configuring and Using SNMP

Part 2. Using the SNMP Subagent DPI API

Part 3. Appendixes

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, U.S.A.

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 the software interoperability coordinator. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Address your questions to:

IBM Corporation
Software Interoperability Coordinator
3605 Highway 52 N
Rochester, MN 55901-7829 USA

This publication could contain technical inaccuracies or typographical errors.

This publication may refer to products that are announced but not currently available in your country. This publication may also refer to products that have not been announced in your country. IBM makes no commitment to make available any unannounced products referred to herein. The final decision to announce any product is based on IBM's business and technical judgment.

This publication 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.

This publication contains small programs that are furnished by IBM as simple examples to provide an illustration. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. All programs contained herein are provided to you "AS IS". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED.

Programming Interface Information

This book is intended to help applicaton programmers use the SNMP function of the IBM OS/400 licensed program. This book documents General-Use Programming Interface and Associated Guidance Information.

General-Use programming interfaces allow the customer to write programs that obtain the services of the OS/400 licensed program.

Trademarks and Service Marks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

ACF/VTAM
Advanced Peer-to-Peer Networking
AIX
AnyNet
Application System/400
APPN
AS/400
C/400
IBM
ILE
NetView
Operating System/400
OS/400
PS/2
RISC System/6000
RS/6000
VTAM
400

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

About Simple Network Management Protocol (SNMP) Support (SC41-5412)

This book is intended for the administrator who is responsible for configuring SNMP related systems management function in OS/400. It is also a guide for the programmer who intends to write SNMP managing applications or SNMP subagents.

SNMP support in the AS/400 system includes the following:

- Configuring and using the SNMP agent
- Configuring and using TRAP manager support
- Using client inventory management
- Using the SNMP subagent DPI API.

Using this book, the AS/400 programmer can:

- Configure the AS/400 system to use SNMP support
- Create SNMP subagents that can respond to SNMP managing applications
- Create managing applications that can request information from appropriate SNMP subagents.

You should be familiar with the following to use the information in this book:

- AS/400 programming terminology. You should also be familiar with the terminology of the host system.
- Data communications concepts.

Prerequisite and Related Information

For information about other AS/400 publications (except Advanced 36), see either of the following:

- The *Publications Reference* book, SC41-5003, in the AS/400 Softcopy Library.
- The *AS/400 Information Directory*, a unique, multimedia interface to a searchable database that contains descriptions of titles available from IBM or from selected other publishers. The *AS/400 Information Directory* is shipped with the OS/400 operating system at no charge.

Information Available on the World Wide Web

More AS/400 information is available on the World Wide Web. You can access this information from the AS/400 home page, which is at the following uniform resource locator (URL) address:

`http://www.as400.ibm.com`

Select the Information Desk, and you will be able to access a variety of AS/400 information topics from that page.

Chapter 1. Introduction to SNMP for OS/400

This chapter describes the Simple Network Management Protocol (SNMP), SNMP components (manager, agent, subagent), and Management Information Bases (MIBs) related to SNMP. The implementation of SNMP for OS/400 is also discussed.

Definition of Simple Network Management Protocol (SNMP)

Simple Network Management Protocol (SNMP) is an industry standard management protocol that originated for TCP/IP networks. SNMP is described by a series of Request for Comments (RFCs) that specifies and structures the information that is exchanged between managing and managed systems. SNMP is used predominately in TCP/IP networks. However, OS/400 AnyNet support allows OS/400 SNMP support to be used in a SNA network.

Definition of SNMP Agent

SNMP agents reside on systems that are managed. The agent receives requests to either retrieve or change management information by referencing **MIB objects**. Management Information Base (MIB) objects are units of information that provide information about the system and the network to the managing system. MIB objects are referenced by the agent whenever a valid request from an **SNMP manager** is received.

Definition of SNMP Manager

An SNMP manager refers to a system that runs a managing application or suite of applications. These applications depend on MIB objects for information that resides on the managed systems. Managers generate requests for this MIB information, and an SNMP agent on the managed system responds to these requests. A request can either be the retrieval or modification of MIB information.

By accessing the MIB objects, the SNMP agent allows configuration, performance, and problem management data to be managed by the SNMP manager. This is how the agent makes network and system information available to other systems.

For example, a managing application that displays system descriptions can be running on a NetView for AIX management platform. If a user needs to access the AS/400 description, the managing application constructs a message that requests a MIB object called sysDescr. This request is sent to the AS/400 system where the SNMP agent decodes the message and then retrieves the information from the sysDescr MIB object. The agent constructs a response with this information and sends it back to the managing application. When the application has decoded the response, the SNMP manager can then display the AS/400 description information to the user.

You can use both the SNMP manager APIs and the SNMP trap support to write an SNMP management application on an AS/400 system. You can retrieve and set MIB data by using the SNMP manager APIs. You can monitor for unsolicited SNMP trap messages by using the SNMP trap support. For a more detailed description of the OS/400 SNMP manager APIs, along with sample source code, see “Simple Network Management Protocol (SNMP) Manager APIs” in the book *System API Reference: UNIX-Type APIs*.

Definition of SNMP Subagents

SNMP agents have predefined MIB objects that they can access. This limits the manager in regards to the type of information that it can request. The need to overcome this limitation brought about the introduction of subagents. A **subagent** allows the dynamic addition of other MIB objects without the need to change the agent. At the same time, SNMP managers are not impacted by these dynamic additions because they continue to work directly with the SNMP agent. Therefore, an SNMP management application can choose to work as if the MIB objects came directly from the SNMP agent.

MIB Objects Supported by the AS/400 SNMP Agent

MIB objects are units of managed information that specifically describe an aspect of a system such as the system name, hardware number, or communication configuration. A collection of related MIB objects is defined as a MIB. The following are the main MIBs that are supported by OS/400.

Standard RFC MIBs

- MIB II (RFC1213)
- Ethernet-like (RFC1398)
- FDDI (RFC1285)
- Frame Relay (RFC1315)
- Token Ring (RFC1231)

IBM enterprise MIBs

- APPN MIB
- Client Management MIB
- NetView for AIX subagent MIB
- DPI 2.0 (RFC1592)
- SNMP subagent MIB

All of the IBM enterprise MIBs except for DPI 2.0 are provided in QSYS/QANMMIB. The DPI 2.0 MIB is not used in OS/400. For more specific information concerning the supported MIBs associated with SNMP, refer to Appendix A, "OS/400 SNMP Agent Set Processing and Supported SNMP MIBs."

Figure 1-1 provides a general overview of the SNMP components and how they reference MIBs. Responses, requests, and traps are discussed in the next section.

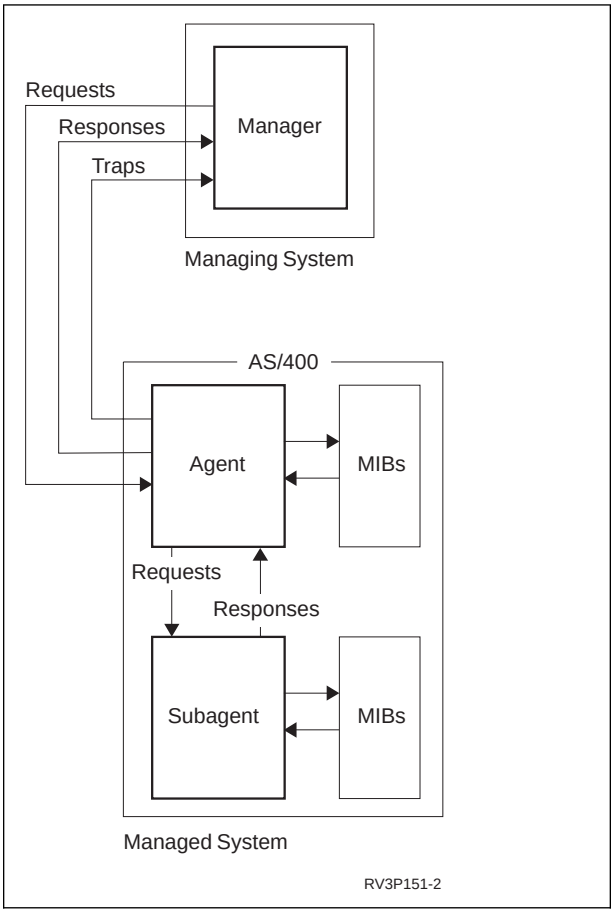


Figure 1-1. SNMP Overview

Supported SNMP Protocol Elements

The OS/400 SNMP and MIB implementations are based on the following Internet RFCs:

- RFC1157 Simple Network Management Protocol (SNMP)
- RFC1155 Structure and Identification of Management Information for TCP/IP-based Internets
- RFC1212 Concise MIB Definition
- RFC1213 Management Information Base for Network Management of TCP/IP-based Internets: MIB-II
- RFC1215 Convention for Defining Traps for Use with SNMP
- RFC1720 IAB Official Protocol Standards
- RFC1700 Assigned Numbers

- RFC1592 Distributed Protocol Interface (DPI) 2.0

The type of protocol used by SNMP to pass data between the SNMP manager and the SNMP agent is a request/response protocol. The manager makes a request for MIB object information and the agent responds to that request. SNMP uses five protocol operations to implement this type of protocol:

1. **GET** - Request to inspect one or more MIB objects
2. **GETNEXT** - Request to inspect the next MIB object
3. **SET** - Request to alter the value of one or more MIB objects
4. **RESPONSE** - Response to a **GET**, **GETNEXT**, or **SET** request.
5. **TRAPS** - Unsolicited messages. Including: coldStart, warmStart, linkDown, linkUp, authenticationFailure, EGPNeighborLoss, and enterpriseSpecific.

The **GET**, **GETNEXT**, and **SET** requests are issued by the SNMP manager to the SNMP agent. The **RESPONSE** request is issued by the agent to the manager. **TRAPS** are sent from the agent to one or more managers as unsolicited messages.

As an example: an SNMP manager requests configuration information for a particular system. The manager formats this request in a **GET** protocol data unit (PDU)[1] and transmits the request to the agent using a communication service. After the manager's request has been received, the agent packages the requested MIB object information in a **RESPONSE** PDU and transmits it back to the manager.

[1] A protocol data unit (PDU) is a protocol unit of work.

Part 1. Configuring and Using SNMP

Chapter 2. Configuring the OS/400 SNMP Agent

This chapter describes how to configure the OS/400 SNMP agent. SNMP agent configuration on the AS/400 consists of the following sections:

- **Communities:** You need to describe how communities are used so that the SNMP agent can determine which SNMP manager requests to honor.
- **SNMP Agent Attributes:** You need to describe the values and options that are used by the SNMP agent, including the destination SNMP managers for traps generated by the agent.

Communities

The relationship between a SNMP agent and one or more SNMP managers is controlled by using communities. Each community consists of the following elements:

community name
Specifies the name of a community. The SNMP manager must specify a community name that is recognizable to the SNMP agent so the agent can honor the manager's request.

object access specification
Specifies what operations the SNMP managers are allowed to perform.

list of SNMP managers' IP addresses
The SNMP manager must specify an IP address that is recognizable to the SNMP agent so that the agent can honor the manager's request.

Community Naming Conventions

A community name consists of one or more characters. Typically, community names consist of readable characters. They can also consist of non-readable characters. These non-readable characters can take on any value between X'00' and X'FF'. Because of possible non-readable characters, the Add Community for SNMP (ADDCOMSNMP), Change Community for SNMP (CHGCOMSNMP), and Remove Community for SNMP (RMVCOMSNMP) commands have an additional parameter; translate community name (ASCIICOM). ASCIICOM determines whether a community name is translated to ASCII characters before it is compared with the community name in a received SNMP manager's request. The ASCIICOM parameter has two possible values, *YES and *NO. Most of the time ASCIICOM(*YES) is specified. If a community specifies ASCIICOM(*YES), the SNMP agent translates the community name from EBCDIC to ASCII characters before comparing it with the community name specified in the SNMP manager request. This feature ensures compatibility between an AS/400 system (an EBCDIC system) and ASCII managers. ASCIICOM(*NO) is specified only if the managing system sends community names in EBCDIC or if the community name consists of one or more non-readable characters. When ASCIICOM(*NO) is specified, translation does not occur.

The combination of the community name and the ASCIICOM value identifies a community in OS/400. For example, community 'public' ASCIICOM(*YES) and community 'public' ASCIICOM(*NO) are two different communities.

Object Access Specification

Object access for a community is specified on the ADDCOMSNMP and CHGCOMSNMP commands by the OBJACC parameter. The OBJACC parameter contains one of four values:

***READ**
managers in this community can access all readable objects for all MIB groups.

***WRITE**
managers in this community can access all readable MIB objects and can change all changeable MIB objects.

***NONE**
managers in this community cannot access any MIB objects.

Note: This value can be used to temporarily deny access to managers in a community without removing the community.

***SNMPATR**
the community is to use the object access value specified in the SNMP attributes.

Manager List

The list of SNMP managers in the community is specified by the manager Internet address (INTNETADR) parameter on the ADDCOMSNMP and CHGCOMSNMP commands. This list consists of the IP addresses of the managers in the community.

A special value of *ANY is supported on the INTNETADR parameter for the ADDCOMSNMP and CHGCOMSNMP commands. This value indicates that all SNMP managers in the network could be part of this community.

OS/400 Community Attributes

For OS/400, a community has two additional attributes:

- LOGSET (log set request) - controls the logging of **SET** requests in journal QSNMP in library QUSRSYS, and
- LOGGET (log get request) - controls the logging of **GET** and **GETNEXT** requests in journal QSNMP in library QUSRSYS.

Each of the community logging parameters has three possible values:

***YES**
manager's requests are logged in the journal.

***NO**
manager's requests are not logged.

***SNMPATR**
the logging value specified in the SNMP attributes is to be used for that community.

For more information on SNMP logging, refer to Appendix B, “Journal for SNMP Logging” on page B-1.

Procedures for Configuring a Community

New communities are defined and added to the OS/400 SNMP agent configuration information by either:

- Issuing the ADDCOMSNMP command
- Selecting option 1 on the Work with Communities for SNMP display. This work with display is shown by selecting option 2 after entering the Configure TCP/IP SNMP (CFGTCPSNMP) command.

Existing communities are changed by either:

- Issuing the CHGCOMSNMP command
- Selecting option 2 on the Work with Communities for SNMP display. This work with display is shown by selecting option 2 after entering the Configure TCP/IP SNMP (CFGTCPSNMP) command.

Existing communities can be removed by either

- Issuing the RMVCOMSNMP command
- Selecting option 4 on the Work with Communities for SNMP display. This work with display is shown by selecting option 2 after entering the Configure TCP/IP SNMP (CFGTCPSNMP) command.

Existing communities can be listed by using the Work with Communities for SNMP display. This display is shown by selecting option 2 after entering the CFGTCPSNMP command. The attributes of a single community can be shown by selecting option 5 on the Work with Communities for SNMP display.

SNMP Agent Attributes

SNMP agent attributes are specified on the AS/400 by either:

- Running the CHGSNMPA command
- Selecting option 1 on the Configure TCP/IP SNMP menu display that is shown by running the CFGTCPSNMP command.

The following are parameters which are used when specifying an SNMP agent on the AS/400 system.

System Contact Parameter

This parameter corresponds to the sysContact MIB object of MIB-II. When *CNTINF is specified, the system contact is set to the local system contact value specified on the Work with Support Contact Information (WRKCNTINF) command (option 2 - Work with local service information).

This value can be changed by a manager that is part of a community that specifies *WRITE access. It can also be read by a manager that is part of a community that specifies *READ or *WRITE access.

System Location Parameter

This parameter corresponds to the sysLocation MIB object of MIB-II. When *CNTINF is specified, the system location is set to the local system location value specified on the WRKCNTINF command (option 2 - Work with local service information). This value can be changed by a manager that is part of a community that specifies *WRITE access. It can also be read by a manager that is part of a community that specifies *READ or *WRITE access.

Send Authentication Traps Parameter

This parameter corresponds to the snmpEnableAuthenTraps MIB object of MIB-II. Send Authentication Traps has a value of either *YES or *NO.

***YES**
An authenticationFailure trap is sent to the managers defined in the trap managers attribute each time a request is received from an SNMP manager with a community name that is unknown to the SNMP agent.

***NO**
No authenticationFailure traps will be sent.

This value can be changed by a manager that is part of a community that specifies *WRITE access. It can also be read by a manager that is part of a community that specifies *READ or *WRITE access.

Automatic Start Parameter

This parameter controls whether the SNMP agent is started when the Start TCP/IP (STRTCP) command runs.

***YES**
The SNMP agent is started.

***NO**
The SNMP agent is not started.

Note: The Start TCP Server (STRTCPSVR) command must be used to start the SNMP agent in this case.

Object Access and the Logging Requests Parameters

The object access, log set request, and the log get request function the same way as their community parameter counterparts.

Log Traps Parameter

This parameter has the following values:

***YES**
A log entry is added to the QSNMP journal in the QUSRSYS library when the SNMP agent sends a trap.

***NO**
Trap logging does not occur.

Trap Manager Parameter

This parameter specifies which SNMP managers are sent the OS/400 SNMP agent's generated traps. Depending on the length of each element in the manager's list, up to 300 managers can be specified. Each element of the manager's list consists of three parts:

- The contents of the manager's IP address.

  Note: This address must be unique.

- The community name that the agent puts in the trap before it is sent to the manager.

  Note: This name is completely independent from the community names that the agent uses to verify requests from a manager.

- An indication of whether the community name should be translated to ASCII characters before it is placed into the trap. The same translation rules used for community names apply.

Pre-Configured Community Attributes

The OS/400 system is shipped with the SNMP agent pre-configured and with one pre-configured community.

community name	public
ASCIICOM	*YES
INTNETADR	*ANY
OBJACC	*READ
LOGSET	*NO
LOGGET	*NO

The pre-configured SNMP agent has the following attributes:

system contact	*NONE
system location	*NONE
send authentication traps	*YES
autostart	*NO
object access	*READ
log set requests	*NO
log get requests	*NO
log traps	*NO
trap managers	*NONE

Configuring Your System for SNMP

The following commands start and stop the SNMP agent. The OS/400 subagent starts and ends in parallel with the agent.

STRTCPSVR (*SNMP) - Starts TCP/IP SNMP agent

ENDTCPSVR (*SNMP) - Ends TCP/IP SNMP agent

The following commands change the SNMP configuration:

- Add Community for SNMP - ADDCOMSNMP
- Configure TCP/IP for SNMP - CFGTCPSNMP
- Change Community for SNMP - CHGCOMSNMP
- Change SNMP Attributes - CHGSNMPA
- Remove Community for SNMP - RMVCOMSNMP

The following are descriptions and examples of the preceding configuration commands. Of course, specific parameters would depend on your system's specific requirements. For additional information on these commands, see Appendix D, “CL Commands for AS/400 SNMP” on page D-1.

ADDCOMSNMP (Add Community for SNMP) Command

The Add Community for SNMP (ADDCOMSNMP) command allows the user to add an SNMP community profile to the SNMP agent configuration file. An SNMP agent uses a community profile to determine whether or not to honor a request sent by an SNMP manager.

This example adds community 'ourcommunity' to the SNMP agent configuration file. The SNMP managers with the Internet addresses '8.6.5.4' and '8.6.5.3' are the only managers in the community (with set capability) and are able to change MIB objects.

```
ADDCOMSNMP COM('ourcommunity')  OBJACC(*WRITE)
  INTNETADR('8.6.5.4' '8.6.5.3')
```

Note: It is important to check to see if the manager has multiple IP addresses. If the manager has multiple addresses, use the same IP address that the manager uses to send PDUs or enter all the hosts IP addresses for the manager's community.

CHGCOMSNMP (Change Community for SNMP) Command

The Change Community for SNMP (CHGCOMSNMP) command allows the user to change an SNMP community profile in the SNMP agent configuration file.

This example changes community 'ourcommunity' in the SNMP agent configuration file.

```
CHGCOMSNMP COM('ourcommunity')  OBJACC(*READ)
```

RMVCOMSNMP (Remove Community for SNMP) Command

The Remove Community for SNMP (RMVCOMSNMP) command allows the user to remove an SNMP community profile from the SNMP agent configuration file.

This command removes community ('ourcommunity') from the SNMP agent configuration file.

```
RMVCOMSNMP  COM('ourcommunity')
```

CHGSNMPA (Change SNMP Attributes) Command

The Change SNMP Attributes (CHGSNMPA) command allows the user to change values and options used by the SNMP agent.

This example changes the system contact information and specifies that the SNMP agent should not start when the STRTCP command runs.

```
CHGSNMPA SYSCONTACT('Joe Smith, telephone 555-1212')
  AUTOSTART(*NO)
```

CFGTCPSNMP (Configure TCP/IP for SNMP) Command

The Configure TCP/IP for SNMP (CFGTCPSNMP) command is used to display a menu that allows a user to define or change the SNMP configuration.

This command presents the following series of menus that are used to define or change your SNMP configuration.

```
CFGTCPSNMP
```

```
                          Configure TCP/IP SNMP
                                                        System:   ENDAS003
Select one of the following:

     1. Change SNMP attributes
     2. Work with communities for SNMP









Selection or command
===> _____________________________________________________________________
F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel
(C) COPYRIGHT IBM CORP.
```

Figure 2-1. CFGTCPSNMP main menu

Option 1 of the Configure TCP/IP SNMP menu displays the Change SNMP Attributes (CHGSNMPA) command in prompt mode.

Option 2 causes the Work with Communities for SNMP display to be shown.

Work with Communities for SNMP Display: Option 2 of the Configure TCP/IP SNMP menu causes the following display to be shown:

```
                        Work with Communities for SNMP
                                                        System:   ENDAS003
Type options, press Enter.
  1=Add   2=Change   4=Remove   5=Display

Option  Community Name
  _     Payroll
  _     Finance
  _     Engineering
  _     Accounting
  _
  _
  _
  _
  _
  _
  _
  _
  _
  _
  _
More...
 F3=Exit   F5=Refresh   F6=Print list   F12=Cancel   F17=Top   F18=Bottom
```

Figure 2-2. Work with Communities for SNMP display

Option 1 of the Work with Communities for SNMP display shows the ADDCOMSNMP command in prompt mode.

Option 2 of the Work with Communities for SNMP display shows the CHGCOMSNMP command in prompt mode.

Option 4 is used to remove a community. It causes the Confirm Remove of Communities for SNMP display to be shown.

Option 5 is used to display detailed community information. It causes the Display Community for SNMP display to be shown.

Pressing F6 causes detailed community information to be printed for all communities.

Confirm Remove of Communities for SNMP Display: Option 4 causes the Work with Communities for SNMP display to be shown:

```
                    Confirm Remove of Communities for SNMP
                                                          System:   ENDAS003
Press Enter to confirm your choices for 4=Remove.
Press F12 to return to change your choices.

Option  Community Name
  4     badcommunity
  4     badcommunity2
  4     badcommunity3









Bottom
F12=Cancel
```

Figure 2-3. Confirm Remove of Communities for SNMP display

Pressing enter causes the RMVCOMSNMP command to be run for each entry in the list.

Display Community for SNMP

Display: Option 5 causes the Work with Communities for SNMP display to be shown:

```
                          Display Community for SNMP
                                                          System:   ENDAS003
Community name . . . . . . . . . . . . . . :   public



Object access  . . . . . . . . . . . . . . :   *READ
Log set requests . . . . . . . . . . . . . :   *YES
Log get requests . . . . . . . . . . . . . :   *NO
Translate community name . . . . . . . . . :   *YES

Manager             Manager             Manager             Manager
Internet            Internet            Internet            Internet
Address             Address             Address             Address
8.145.150.129       8.145.150.134       8.145.150.139       8.145.150.144
8.145.150.130       8.145.150.135       8.145.150.140
8.145.150.131       8.145.150.136       8.145.150.141
8.145.150.132       8.145.150.137       8.145.150.142
8.145.150.133       8.145.150.138       8.145.150.143

Bottom
Press Enter to continue.

F3=Exit   F12=Cancel
```

Figure 2-4. Display Community for SNMP display

Chapter 3. SNMP Manager Enablement

This chapter describes how the SNMP manager is enabled for OS/400.

The OS/400 SNMP manager provides a rudimentary base for SNMP management applications. The OS/400 SNMP manager consists of the following functions:

- SNMP manager application program interface (API)
- Trap manager

SNMP Manager APIs

The SNMP manager APIs provide SNMP management applications the ability to perform SNMP management functions (GET, GETNEXT, and SET) to local or remote SNMP agents. For additional information on these APIs, see “Simple Network Management Protocol (SNMP) Manager APIs” in the book *System API Reference: UNIX-Type APIs*.

Trap Manager

The trap manager receives traps, parses traps, and then enqueues the traps to an internal queue. If configured to do so, the trap manager then sends the traps to other management destinations as they are configured in the SNMP agent on the AS/400 system. This type of function gives users the ability to decide whether to forward traps that are received from other nodes. The traps are forwarded through the SNMP agent that generates and sends the actual trap PDU. Therefore, traps are sent to all managers that are configured in the SNMP agent's trap manager attributes. Trap forwarding is configurable using a CL command interface.

Note: Use care when configuring the trap destinations to avoid trap loops. (Loops are caused when two or more trap managers are configured to forward traps to each other.)

The following is a list of trap manager commands:

- STRTRPMGR - Start trap manager
- ENDTRPMGR - End trap manager

STRTRPMGR (Start Trap Manager) Command

The Start Trap Manager (STRTRPMGR) command allows the user to start the OS/400 SNMP trap manager job. An optional Forward Trap parameter may be specified. This parameter enables all traps received on port 162 of this system to be forwarded to the SNMP managers listed in the SNMP agent's trap manager attribute. When configuring the SNMP agent's trap manager attribute and specifying forwarding on in the STRTRPMGR command, refer to Appendix C, “Problem Analysis for SNMP” on page C-1.

This example starts the trap management job. Traps that are received by the trap manager are enqueued and forwarded.

```
STRTRPMGR FWDTRP(*YES)
```

The other valid value for the FWDTRP parameter is *NO. For additional information on the STRTRPMGR command, see “STRTRPMGR (Start Trap Manager) Command” on page D-13.

ENDTRPMGR (End Trap Manager) Command

The End Trap Manager (ENDTRPMGR) command allows the user to end the OS/400 SNMP trap manager job.

This example ends the trap management job.

```
ENDTRPMGR
```

SNMP Trap Support

You can monitor for unsolicited SNMP trap messages by using the SNMP trap support. These trap messages may contain helpful data for managing a network.

By using the OS/400 SNMP manager, it is possible to deliver SNMP traps to data queues. All traps that are received on an AS/400 system can be routed to user-defined data queues. For additional information on configuring and using trap support, see “SNMP Trap Support” in the book *System API Reference: UNIX-Type APIs*.

Chapter 4. Client Inventory Management

Client Inventory Management can be used to provide the host with information about the client. If a client is configured to send SNMP traps to a managing AS/400, the host collects hardware, software, and connectivity information from the managed client. This chapter describes the Client Inventory Management and lists the Client Management database formats.

Client Inventory Management

OS/400 gathers information about personal computer clients that are attached to the AS/400. The information is stored in AS/400 database files to provide a comprehensive inventory of personal computer assets. An example of the type of information available is the following:

- Directory - connectivity, access, and management information
- Software - installed software identification and change information
- Hardware - disk drive, memory, hard file

Traps that are received by an AS/400 system are processed differently for new clients than traps that are received for existing clients. New client's traps are processed immediately and are added for inventory management. Hardware and software information about this client is retrieved and stored in the database.

For existing clients, hardware and software information is only refreshed if this information is more than 30 days old. 30 days is the default value. The default value can be changed to any value between 1 and 365 days. Any other value results in default of 30 days. A data area QZCAREFI in library QUSRSYS should be created to change the refresh interval. For example:

```
CRTDTAARA DTAARA(QUSRSYS/QZCAREFI)
  TYPE(*DEC) LEN(2 0) VALUE(99)
  TEXT('Refresh Interval')
```

This example sets the refresh interval at 99 days. The data area must be of type decimal (integer value). Using this example, a refresh interval of 99 days is used for all clients that the AS/400 system is managing.

Using the Client Management MIB, users can write application programs that perform resource, asset, license, and network management functions by accessing the information in the Client Management Databases.[1] The Client Management Databases, which are a set of physical and logical files, represent the client information that was gathered using SNMP **GET** and **GETNEXT** requests when the client was connected to an AS/400. The database structure is based on the MIB objects that are being shadowed from the client.

APIs are provided to manage the clients in the Client Information Database. Using these APIs you can add, change, or remove client information that is stored on the client.

Client Software Management Database Formats

The following are the database formats used when querying and writing applications that access client management information. All of these database files are shipped with OS/400 in library QSYS. The files are copied into the QUSRSYS library when the first client is discovered and stored in the database. All client information is stored in the database files in the QUSRSYS library. The files in the QSYS library are for recovery purposes.

[1] The user is not able to write information to the databases. They are able to write programs that access the databases.

QAZCADEV

```
***   Start of specifications   *********************************
*
*  File Name:  qazcadev
*  File Type:  physical
*
*  Function:  Database file where the client device
*             information is maintained.  The data is obtained
*             from the HOST MIB.
*
*         ZCAIDX --> Client Index
*         ZCADEV --> Device Index
*         ZCATYP --> Device Type
*         ZCADID --> Device ID
*         ZCASTT --> Device Status
*         ZCAERR --> Device Errors
*         ZCADSC --> Device Description
*
***   End of Specifications   *********************************
                                    UNIQUE
          R QZCADEVR
            ZCAIDX        12H
            ZCADEV         9B
            ZCATYP         9B
            ZCADID       128A         VARLEN(128)
            ZCASTT         9B
            ZCAERR         9B
            ZCADSC        64A         VARLEN(64)
          K ZCAIDX
          K ZCADEV
```

QAZCADIR

```
***   Start of specifications   *********************************
*
*  File Name:  qazcadir
*  File Type:  physical
*
*  Function:  Database file where the basic client directory
*             information is maintained.
*
*         ZCBIDX --> Client Index
*         ZCBCLT --> Client ID
*         ZCBDSC --> Client Description
*         ZCBCMN --> Community
*         ZCBIPA --> IP Address
*         ZCBCPN --> CP NetID
*
***   End of Specifications   *********************************
                                    UNIQUE
          R QZCADIRR
            ZCBIDX        12H
            ZCBCLT       255A         VARLEN(255)
            ZCBDSC       255A         VARLEN(255)
            ZCBCMN       255A         VARLEN(255)
            ZCBIPA        15A         VARLEN(15)
            ZCBCPN        17A         VARLEN(17)
          K ZCBIDX
```

QAZCADSK

```
***   Start of specifications   *********************************
*
*  File Name:  qazcadsk
*  File Type:  physical
*
*  Function:  Database file where the client disk storage
*             information is maintained.  The data is obtained
*             from the HOST MIB.
*
*         ZCCIDX --> Client Index
*         ZCCDEV --> Device Index
*         ZCCACC --> Disk Access
*         ZCCMED --> Disk Media
*         ZCCRMV --> Disk Media Remove
*         ZCCCAP --> Disk Capacity
*
***   End of Specifications   *********************************
                                    UNIQUE
          R QZCADSKR
            ZCCIDX        12H
            ZCCDEV         9B
            ZCCACC         9B
            ZCCMED         9B
            ZCCRMV         9B
            ZCCCAP         9B
          K ZCCIDX
          K ZCCDEV
```

QAZCAFS

```
***   Start of specifications   *********************************
*
*  File Name:  qazcafs
*  File Type:  physical
*
*  Function:  Database file where the client file system
*             information is maintained.  The data is obtained
*             from the HOST MIB.
*
*         ZCDIDX --> Client Index
*         ZCDFS  --> File System Index
*         ZCDLMP --> Mount Point
*         ZCDRMP --> Remote Mount
*         ZCDTYP --> Type
*         ZCDACC --> Access
*         ZCDBOT --> Bootable
*         ZCDSTG --> Storage Index
*         ZCDFBK --> Full Backup
*         ZCDPBK --> Partial Backup
*
***   End of Specifications   *********************************
                                    UNIQUE
          R QZCAFSR
            ZCDIDX        12H
            ZCDFS          9B
            ZCDLMP       128A         VARLEN(128)
            ZCDRMP       128A         VARLEN(128)
            ZCDTYP         9B
            ZCDACC         9B
            ZCDBOT         9B
            ZCDSTG         9B
            ZCDFBK          Z
            ZCDPBK          Z
          K ZCDIDX
          K ZCDFS
```

QAZCAMSC

```
***   Start of specifications   *********************************
*
*  File Name:  qazcamsc
*  File Type:  physical
*
*  Function:  Database file where the basic client directory
*             information is maintained
*
*         ZCBIDX --> Client Index
*         ZCBMEM --> Memory size
*         ZCBUPT --> Up Time
*         ZCBSTT --> System Status
*         ZCBMBI --> Support MIBII
*         ZCBMBH --> Support HOST MIB
*         ZCBMBA --> Support APPN MIB
*         ZCBMBX --> Support Extensions
*         ZCBHDW --> Hardware Refresh
*         ZCBSFW --> Software Refresh
*         ZCBCON --> Contact
*         ZCBLOC --> Location
*         ZCBTYP --> Machine Type
*         ZCBMDL --> Machine Model
*         ZCBUSR --> User Profile
*         ZCBOWN --> Owner
*         ZCBPHN --> Owner Phone
*         ZCBOFC --> Office
*
***   End of Specifications   *********************************
```

```
                                    UNIQUE
          R QZCAMSCR
            ZCBIDX        12H
            ZCBMEM         9B
            ZCBUPT         9B
            ZCBSTT         9B
            ZCBMBI         9B
            ZCBMBH         9B
            ZCBMBA         9B
            ZCBMBX         9B
            ZCBHDW          Z
            ZCBSFW          Z
            ZCBCON       255A         VARLEN(255)
            ZCBLOC       255A         VARLEN(255)
            ZCBTYP         4A         VARLEN(4)
            ZCBMDL         3A         VARLEN(3)
            ZCBUSR        10A         VARLEN(10)
            ZCBOWN        32A         VARLEN(32)
            ZCBPHN        32A         VARLEN(32)
            ZCBOFC        32A         VARLEN(32)
          K ZCBIDX
```

QAZCANET

```
***   Start of specifications   *********************************
*
*  File Name:  qazcanet
*  File Type:  physical
*
*  Function:  Database file where the client network
*             information is maintained.  The data is obtained
*             from the HOST MIB.
*
*         ZCEIDX --> Client Index
*         ZCEDEV --> Device Index
*         ZCENIF --> Network IFIndex
*
***   End of Specifications   *********************************
                                    UNIQUE
          R QZCANETR
            ZCEIDX        12H
            ZCEDEV         9B
            ZCENIF         9B
          K ZCEIDX
          K ZCEDEV
```

QAZCAPRC

```
***   Start of specifications   *********************************
*
*  File Name:  qazcaprc
*  File Type:  physical
*
*  Function:  Database file where the client processor
*             information is maintained.  The data is obtained
*             from the HOST MIB.
*
*         ZCFIDX --> Client Index
*         ZCFDEV --> Device Index
*         ZCFLOD --> Processor Load
*         ZCFFRM --> Processor Licensed Internal Code
*
***   End of Specifications   *********************************
                                    UNIQUE
          R QZCAPRCR
            ZCFIDX        12H
            ZCFDEV         9B
            ZCFLOD         9B
            ZCFFRM       128A         VARLEN(128)
          K ZCFIDX
          K ZCFDEV
```

QAZCAPRT

```
***   Start of specifications   *********************************
*
*  File Name:  qazcaprt
*  File Type:  physical
*
*  Function:  Database file where the client printer
*             information is maintained.  The data is obtained
*             from the HOST MIB.
*
*         ZCGIDX --> Client Index
*         ZCGDEV --> Device Index
*         ZCGSTT --> Status
*         ZCGERR --> Error State
*
***   End of Specifications   *********************************
                                    UNIQUE
          R QZCAPRTR
            ZCGIDX        12H
            ZCGDEV         9B
            ZCGSTT         9B
            ZCGERR         4H         VARLEN(1)
          K ZCGIDX
          K ZCGDEV
```

QAZCAPTN

```
***   Start of specifications   *********************************
*
*  File Name:  qazcaptn
*  File Type:  physical
*
*  Function:  Database file where the client storage partition
*             information is maintained.  The data is obtained
*             from the HOST MIB.
*
*         ZCHIDX --> Client Index
*         ZCHDEV --> Device Index
*         ZCHPTN --> Partition Index
*         ZCHSIZ --> Size
*         ZCHFSX --> FS Index
*         ZCHLBL --> Label
*         ZCHID  --> Id
*
***   End of Specifications   *********************************
                                    UNIQUE
          R QZCAPTNR
            ZCHIDX        12H
            ZCHDEV         9B
            ZCHPTN         9B
            ZCHSIZ         9B
            ZCHFSX         9B
            ZCHLBL       128A         VARLEN(128)
            ZCHID        128H         VARLEN(128)
          K ZCHIDX
          K ZCHDEV
          K ZCHPTN
```

QAZCADRL

```
***   Start of specifications   *********************************
*
*  File Name:  qazcadrl
*  File Type:  logical
*
*  Function:  Database file where the basic client directory
*             information is maintained
*
*         ZCBIDX --> Client index
*         ZCBCLT --> Client ID
*         ZCBDSC --> Client Description
*         ZCBCMN --> Community
*         ZCBIPA --> IP Address
*         ZCBCPN --> CP NetID
*
***   End of Specifications   *********************************
                                    UNIQUE

          R QZCADRLR                PFILE(QSYS/QAZCADIR)
            ZCBCLT       255A         VARLEN
            ZCBIDX        12H
            ZCBDSC       255A         VARLEN
            ZCBCMN       255A         VARLEN
            ZCBIPA        15A         VARLEN
            ZCBCPN        17A         VARLEN
          K ZCBCLT
```

QAZCASFW

```
***   Start of specifications   *********************************
*
*  File Name:  qazcasfw
*  File Type:  physical
*
*  Function:  Database file where the client software
*             information is maintained.  The data is obtained
*             from the HOST MIB extensions.
*
*         ZCJIDX --> Client Index
*         ZCJSFW --> Software Index
*         ZCJTYP --> Software Type
*         ZCJSTT --> Software Status
*         ZCJID  --> Software Id
*         ZCJVER --> Software Version
*         ZCJOPT --> Software Option
*         ZCJFTR --> Software Feature
*         ZCJMNF --> Software Manufacture
*         ZCJPTH --> Software Path
*         ZCJDAT --> Software Date
*         ZCJNAM --> Software Name
*         ZCJSN  --> Software SerialNumber
*
***   End of Specifications   *********************************
                                      UNIQUE
          R QZCASFWR
            ZCJIDX        12H
            ZCJSFW         9B
            ZCJTYP         9B
            ZCJSTT         9B
            ZCJID        128A         VARLEN(7)
            ZCJVER        64A         VARLEN(6)
            ZCJOPT        16A         VARLEN(4)
            ZCJFTR        16A         VARLEN(4)
            ZCJMNF        64A         VARLEN(32)
            ZCJPTH       255A         VARLEN(128)
            ZCJDAT          Z
            ZCJNAM        64A         VARLEN(32)
            ZCJSN         64A         VARLEN(32)
          K ZCJIDX
          K ZCJSFW
```

QAZCASFX

```
***   Start of specifications   *********************************
*
*  File Name:  qazcasfx
*  File Type:  physical
*
*  Function:  Database file where the Client software fix
*             information is maintained.  The data is obtained
*             from the HOST MIB extensions.
*
*         ZCKIDX --> Client Index
*         ZCKSFW --> Software Index
*         ZCKSFX --> Software Fix Index
*         ZCKFIX --> Software Fix Id
*
***   End of Specifications   *********************************
                                      UNIQUE
          R QZCASFXR
            ZCKIDX        12H
            ZCKSFW         9B
            ZCKSFX         9B
            ZCKFIX        16A         VARLEN(7)
          K ZCKIDX
          K ZCKSFW
          K ZCKFIX
```

QAZCASTG

```
***   Start of specifications   *********************************
*
*  File Name:  qazcastg
*  File Type:  physical
*
*  Function:  Database file where the client storage
*             information is maintained.  The data is obtained
*             from the HOST MIB.
*
*         ZCIIDX --> Client index
*         ZCISTG --> Storage index
*         ZCITYP --> Type
*         ZCIDSC --> Description
*         ZCIALU --> Allocation Units
*         ZCISIZ --> Size
*         ZCIUSD --> Used
*         ZCIALF --> Allocation Failure
*
***   End of Specifications   *********************************
                                      UNIQUE
          R QZCASTGR
            ZCIIDX        12H
            ZCISTG         9B
            ZCITYP         9B
            ZCIDSC       128A         VARLEN(128)
            ZCIALU         9B
            ZCISIZ         9B
            ZCIUSD         9B
            ZCIALF         9B
          K ZCIIDX
          K ZCISTG
```

Part 2. Using the SNMP Subagent DPI API

Chapter 5. Introduction to SNMP Distributed Protocol Interface

The Simple Network Management Protocol (SNMP) agent distributed protocol interface (DPI) permits users to dynamically add, delete, or replace management variables in the local Management Information Base (MIB) without requiring you to recompile the SNMP agent.

This chapter describes the SNMP DPI routines that are supported by OS/400. This Application Programming Interface (API) is for the DPI subagent programmer.

For additional information, you may also want to obtain a copy of RFC1592 (SNMP DPI 2.0 RFC).

SNMP Agents and Subagents

SNMP agents are responsible for answering SNMP requests from network management stations. Examples of management requests that are performed on the MIB objects are GET, GETNEXT, and SET.

A subagent extends the set of MIB objects that are provided by the SNMP agent. With the subagent, you define MIB variables useful in your own environment and register them with the SNMP agent.

When the agent receives a request for a MIB variable, it passes the request to the subagent. The subagent then returns a response to the agent. The agent creates an SNMP response packet and sends the response to the remote network management station that initiated the request. The existence of the subagent is transparent to the network management station.

DPI Agent Requests

The SNMP agent can initiate several DPI requests:

- get
- getnext
- set, commit, and undo
- unregister
- close

The SNMP subagent can initiate these DPI requests:

- open, close
- register
- response
- trap

The SNMP subagent communicates with the SNMP agent using these functions:

- connect, disconnect
- receive, wait
- send

The GET, GETNEXT, and SET requests correspond to the SNMP requests that a network management station can make. The subagent responds to a request with a response packet. The response packet can be created using the mkDPIresponse() library routine, which is part of the DPI API library.

COMMIT, UNDO, UNREGISTER, and CLOSE requests are specific SNMP DPI requests.

The subagent normally responds to a request with a RESPONSE packet. For the CLOSE and UNREGISTER requests, the subagent does not need to send a RESPONSE.

SNMP DPI API Source Files

The following source file is provided:

qtossapi.h The public SNMP DPI 2.0 API as provided to the DPI subagent programmer. The DPI subagent code must include this file.

This source file is available in source file H, member QTOSSAPI, in library QSYSINC. (If you cannot locate this file, contact your system support department. The QSYSINC library can be selectively installed at any time.)

Compiling, Linking, and Running a DPI API Application

Programs using the SNMP subagent API must be written in the C language. Use ILE C/400 to compile the program with the Create C Module (CRTCMOD) CL command. After the *MOD object is created, specify BNDSVRPGM(QTOSSAPI) in the Create Program (CRTPGM) CL command. For additional information, see the *ILE C/400 Programmer's Guide*. and the *ILE C/400 Programmer's Reference*.

Additional DPI Information

For information on the SNMP APIs, see "Simple Network Management Protocol (SNMP) Subagent APIs" in the book *System API Reference: UNIX-Type APIs*.

Chapter 6. Subagent Programming Concepts

The DPI API for OS/400 is the 2.0 level of the protocol. This is designed to be highly compatible with SNMPv2 even if the SNMP Agent is SNMPv1. This compatibility allows future upgrades of a subagent implementation for use with a SNMPv2 Agent. Specifically:

- Use only the SNMP Version 2 error codes even if there are definitions for the SNMP Version 1 error codes.
- Implement the SET, COMMIT, UNDO processing as described in the sections that follow.
- For GET requests, use the SNMP Version 2 approach and pass back noSuchInstance value or noSuchObject value if such is the case. Continue to process all remaining varBinds.
- For GETNEXT, use the SNMP Version 2 approach and pass back endOfMibView value if such is the case. Continue to process all remaining varBinds.
- When you are processing a request from the agent (GET, GETNEXT, SET, COMMIT, or UNDO), you are supposed to respond within the time-out period. You can specify the time-out period in the OPEN and REGISTER packets.

  If you fail to respond within that time-out period, the agent will most probably close your DPI connection and then discard your RESPONSE packet if it comes in later. If you can detect that the response is not going to be received in the time period, then you might decide to stop the request and return an SNMP_ERROR_genErr in the RESPONSE.
- You may want to issue an SNMP DPI ARE_YOU_THERE request periodically to ensure that the agent is still connected and still knows about you.
- Generally, SNMP agents and managers use the printable ASCII character set to represent DisplayString values. For additional information on the mkDPIopen() API, see “Simple Network Management Protocol (SNMP) Subagent APIs” in the book *System API Reference: UNIX-Type APIs*.
- If you receive an error RESPONSE on the OPEN packet, you will also receive a DPI CLOSE packet with an SNMP_CLOSE_openError code. In this situation, the agent closes the connection.
- Please realize that DisplayString is only a textual convention. In the SNMP PDU (SNMP packet), the type is just an OCTET_STRING.

  When the type is OCTET_STRING, it is not clear if this is a DisplayString or any arbitrary data. This means that the agent can only know about an object being a DisplayString if the object is included in some sort of a compiled MIB. If it is, the agent will use SNMP_TYPE_DisplayString in the type field of the varBind in a DPI SET packet. When you send a DisplayString in a RESPONSE packet, the agent will handle it as such.

Related Information

RFC1440 through RFC1452 are the SNMP Version 2 RFCs.

GET Processing

The DPI GET packet holds one or more varBinds that the subagent has taken responsibility for.

If the subagent encounters an error while processing the request, it creates a DPI RESPONSE packet with an appropriate error indication in the error_code field. The subagent then sets the error_index to the position of the varBind at which the error occurs. The first varBind is index 1, the second varBind is index 2, and so on. No OID, type, length, or value information needs to be provided in the packet. This is because, by definition, the varBind information is the same as in the request to which this is a response and the agent still has that information.

If there are no errors, the subagent creates a DPI RESPONSE packet in which the error_code is set to SNMP_ERROR_noError (zero) and error_index is set to zero. The packet must also include the OID, type, length, and value of each varBind requested.

When you get a request for a non-existing object or a non-existing instance of an object, you must return a NULL value with a type of SNMP_TYPE_noSuchObject or SNMP_TYPE_noSuchInstance respectively. These two values are not considered errors, so the error_code and error_index should be zero.

The DPI RESPONSE packet is then sent back to the agent.

SET Processing

A DPI SET packet contains the OID, type, length, and value of each varBind requested, plus the value type, value length, and value to be set.

If the subagent encounters an error while processing the request, it creates a DPI RESPONSE packet with an appropriate error indication in the error_code field and an error_index listing the position of the varBind at which the error occurs. The first varBind is index 1, the second varBind is index 2, and so on. No OID, type, length, or value information needs to provided in the packet. This is because, by definition, the varBind information is the same as in the request to which this is a response and the agent still has that information.

If there are no errors, the subagent creates a DPI RESPONSE packet in which the error_code is set to SNMP_ERROR_noError (zero) and error_index is set to zero. No OID, type, length, or value information is needed because the RESPONSE to a SET should contain exactly the same varBind data as the data present in the request. The agent can use the values it already has.

This suggests that the agent must keep state information. This is the case. The subagent keeps state information to make it able to pass the data with a DPI COMMIT or DPI UNDO packet. Because there are no errors, the subagent must have allocated the required resources and prepared itself for the SET. It does not yet carry out the set. That will be done at COMMIT time.

The subagent sends a DPI RESPONSE packet, indicating success or failure for the preparation phase, back to the agent. The agent will issue a SET request for all other varBinds in the same original SNMP request it received. This may be to the same subagent or to one or more different subagents.

Once all SET requests have returned a *no error* condition, the agent starts sending DPI COMMIT packets to the subagents. If any SET request returns an error, the agent sends DPI UNDO packets to those subagents that indicated successful processing of the SET preparation phase.

When the subagent receives the DPI COMMIT packet, all the varBind information will again be available in the packet. The subagent can now carry out the SET request.

If the subagent encounters an error while processing the COMMIT request, it creates a DPI RESPONSE packet with value SNMP_ERROR_commitFailed in the error_code field. An error_index that lists at which varBind the error occurs is also created. The first varBind is index 1, and so on. No OID, type, length, or value information is needed. The fact that a commitFailed error exists does not mean that this error should be returned easily. A subagent should do all that is possible to make a COMMIT succeed.

If there are no errors and the SET and COMMIT have been carried out with success, the subagent creates a DPI RESPONSE packet in which the error_code is set to SNMP_ERROR_noError (zero) and error_index is set to zero. No OID, type, length, or value information is needed.

So far we have discussed a successful SET and COMMIT sequence. However, after a successful SET, the subagent may receive a DPI UNDO packet. The subagent must now undo any preparations it made during the SET processing, such as free allocated memory.

Even after a COMMIT, a subagent may still receive a DPI UNDO packet. This will occur if some other subagent could not complete a COMMIT request. Because of the SNMP requirement that all varBinds in a single SNMP SET request must be changed as if all changes were simultaneous, all committed changes must be undone if any of the COMMIT requests fail. In this case the subagent must try to undo the committed SET operation.

If the subagent encounters an error while processing the UNDO request, it creates a DPI RESPONSE packet with value SNMP_ERROR_undoFailed in the error_code field. An error_index that lists at which varBind the error occurs is also created. The first varBind is index 1, and so on. No OID, type, length, or value information is needed. The fact that an undoFailed error exists does not mean that this error should be returned easily. A subagent should do all that is possible to make an UNDO succeed.

If there are no errors and the UNDO has been successful, the subagent creates a DPI RESPONSE packet in which the error_code is set to SNMP_ERROR_noError (zero) and error_index is set to zero. No OID, type, length, or value information is needed.

GETNEXT Processing

The DPI GETNEXT packet contains the objects on which the GETNEXT operation must be performed. For this operation, the subagent is to return the OID, type, length, and value of the next variable it supports whose (ASN.1) OID lexicographically follows the one passed in the group ID (subtree) and instance ID.

In this case, the instance ID may not be present (NULL) in the incoming DPI packet implying that the NEXT object must be the first instance of the first object in the subtree that was registered.

It is important to realize that a given subagent may support several discontiguous sections of the MIB tree. In that situation, it would be incorrect to jump from one section to another. This problem is correctly handled by examining the group ID in the DPI packet. This group ID represents the reason why the subagent is being called. It holds the prefix of the tree that the subagent had indicated it supported (registered).

If the next variable supported by the subagent does not begin with that prefix, the subagent must return the same object instance as in the request. For example, the group ID and instance ID with a value of SNMP_TYPE_endOfMibView (implied NULL value). This endOfMibView is not considered an error, so the error_code and error_index should be zero. If required, the SNMP agent will call the subagent again, but pass it a different group ID (prefix). This is illustrated in the discussion below.

Assume that there are two subagents. The first subagent registers two distinct sections of the tree: A and C. In reality, the subagent supports variables A.1 and A.2, but it correctly registers the minimal prefix required to uniquely identify the variable class it supports.

The second subagent registers section B, which appears between the two sections that are registered by the first agent.

If a management station begins browsing the MIB, starting from A, the following sequence of queries of the form get-next(group ID,instance ID) would be performed:

```
Subagent 1 gets called:
       get-next(A,none) = A.1
       get-next(A,1)    = A.2
       get-next(A,2)    = endOfMibView

Subagent 2 is then called:
       get-next(B,none) = B.1
       get-next(B,1)    = endOfMibView

Subagent 1 gets called again:
       get-next(C,none) = C.1
```

OPEN Request

After a successful connection is made, a DPI subagent must open a connection with the agent. To do so, the subagent must send a DPI OPEN packet in which these parameters must be specified:

- The maximum time-out value in seconds. The agent is requested to wait this long for a response to any request for an object being handled by this subagent.

  The agent may have an absolute maximum time-out value which will be used if the subagent asks for too large a time-out value. A value of zero can be used to indicate that the agent's own default time-out value should be used. A subagent is advised to use a reasonably short interval of a few seconds. If a specific subtree needs a (much) longer time, a specific REGISTER can be done for that subtree with a longer time-out value.

- The maximum number of varBinds that the subagent is prepared to handle per DPI packet. Specifying 1 would result in DPI Version 1 behavior of one varBind per DPI packet that the agent sends to the subagent. A value of zero means the agent will try to combine up to as many varBinds as are present in the SNMP packet that belongs to the same subtree.
- The character set you want to use. By default, a 0 value. This is the native character set of the machine platform where the agent runs. Since the subagent and agent normally run on the same system or platform, you want to use the native character set, which on many platforms is ASCII.

  If your platform is EBCDIC based, using the native character set of EBCDIC makes it easy to recognize the string representations of the fields, such as the group ID and instance ID. At the same time, the agent will translate the value from printable ASCII to EBCDIC and vice versa for objects that it knows from a compiled MIB to have a textual convention of DisplayString. This fact cannot be determined from the SNMP PDU encoding because in the PDU the object is only known to be an OCTET_STRING.
- The subagent ID. This an ASN.1 Object Identifier that uniquely identifies the subagent. This OID is represented as a null-terminated string using the selected character set.

  For example: `1.3.5.1.2.3.4.5`
- The subagent description. This is a DisplayString that describes the subagent. This is a character string that uses the selected character set.

  For example: `DPI sample subagent Version 2.0`

Once a subagent has sent a DPI OPEN packet to an agent, it should expect a DPI RESPONSE packet that informs the subagent about the result of the request. The packet ID of the RESPONSE packet should be the same as that of the OPEN request to which the RESPONSE packet is the response.

If you receive an error RESPONSE on the OPEN packet, you will also receive a DPI CLOSE packet with an SNMP_CLOSE_openError code. In this situation, the agent closes the connection.

If the OPEN is accepted, the next step is to REGISTER one or more MIB subtrees.

This sequence is depicted in Figure 6-1.

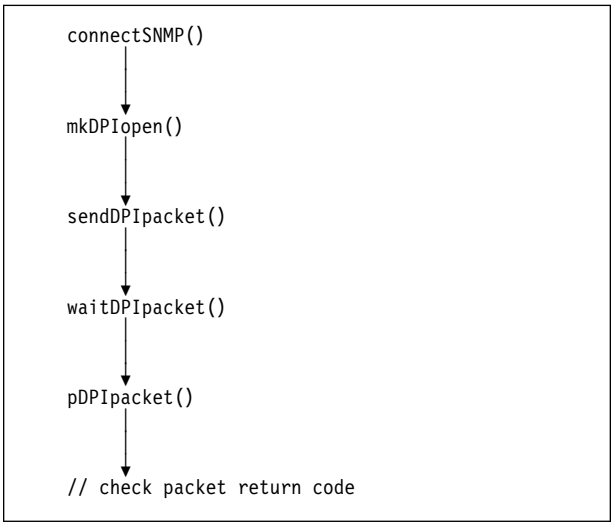


Figure 6-1. Subagent initiation DPI API call sequence.

CLOSE Request

When a subagent is finished and wants to end processing, it should first UNREGISTER its subtrees and then close the connection with the agent. To do so, the subagent must send a DPI CLOSE packet, which specifies a reason for the closing. You should not expect a response to the CLOSE request.

A subagent should also be prepared to handle an incoming DPI CLOSE packet from the agent. In this case, the packet will contain a reason code for the CLOSE request. A subagent does not have to send a response to a CLOSE request. The agent just assumes that the subagent will handle it appropriately. The close takes place regardless of what the subagent does with it.

This sequence is depicted in Figure 6-2 on page 6-5.

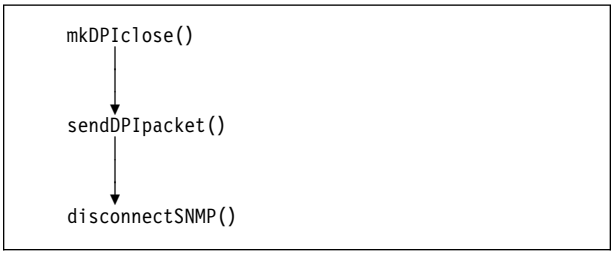


Figure 6-2. Ending a subagent.

REGISTER Request

Before a subagent will receive any requests for MIB variables, it must first register the variables or subtree it supports with the SNMP agent. The subagent must specify a number of parameters in the REGISTER request:

- The subtree to be registered. This is a null-terminated string in the selected character set. The subtree must have a trailing dot.

  For example: `1.3.6.1.2.3.4.5.`

- The requested priority for the registration. The values are:

  -1 Request for the best available priority.

  0 Request for the next best available priority than the highest (best) priority currently registered for this subtree.

  NNN Any other positive value requests that specific priority if available or the next worse priority that is available.

- The maximum time-out value in seconds. The agent is requested to wait this long for a response to any request for an object in this subtree. The agent may have an absolute maximum time-out value which will be used if the subagents ask for too large a time-out value. A value of zero can be used to indicate that the DPI OPEN value should be used for time-out.

- A specification if the subagent wants to do view selection. If it does, the community name from SNMP Version 1 packets will be passed in the DPI GET, GETNEXT, and SET packets.

- A specification if the subagent wants to receive GETBULK packets or if it just prefers that the agent converts a GETBULK into multiple GETNEXT requests.

Once a subagent has sent a DPI REGISTER packet to the agent, it should expect a DPI RESPONSE packet that informs the subagent about the result of the request. The packet ID of the RESPONSE packet should be the same as that of the REGISTER packet to which the RESPONSE packet is the response.

If the response is successful, the error_index field in the RESPONSE packet contains the priority that the agent assigned to the subtree registration.

This sequence is depicted in Figure 6-3.

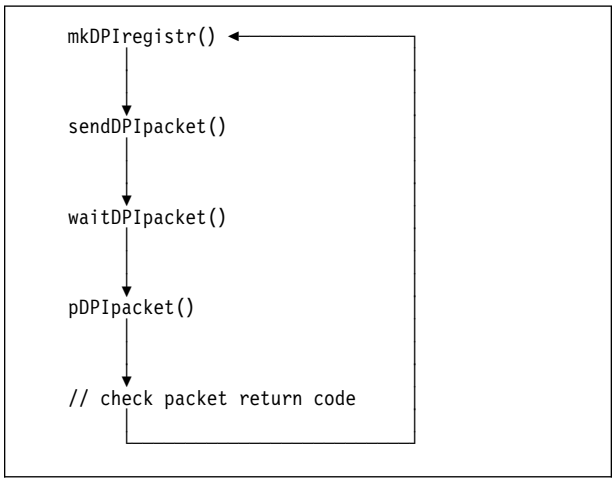


Figure 6-3. Subagent registration DPI API call sequence. (multiple subtrees can be registered.)

UNREGISTER Request

A subagent may unregister a previously registered subtree. The subagent must specify a few parameters in the UNREGISTER request:

- The subtree to be unregistered. This is a null-terminated string in the selected character set. The subtree must have a trailing dot.

  For example: `1.3.6.1.2.3.4.5.`

- The reason for the unregister.

Once a subagent has sent a DPI UNREGISTER packet to the agent, it should expect a DPI RESPONSE packet that informs the subagent about the result of the request. The packet ID of the RESPONSE packet should be the same as

that of the REGISTER packet to which the RESPONSE packet is the response.

A subagent should also be prepared to handle incoming DPI UNREGISTER packets from the agent. In this situation, the DPI packet will contain a reason code for the UNREGISTER. A subagent does not have to send a response to an UNREGISTER request. The agent just assumes that the subagent will handle it appropriately. The registration is removed regardless of what the subagent returns.

TRAP Request

A subagent can request the SNMP agent to generate a trap. The subagent must provide the desired values for the generic and specific parameters of the trap. The subagent may optionally provide a set of one or more OID, type, length, or value parameters that will be included in the trap packet.

It may optionally specify an Enterprise ID (Object Identifier) for the trap to be generated. If a NULL value is specified for the Enterprise ID, the agent will use the subagent Identifier from the DPI OPEN packet as the Enterprise ID to be sent with the trap. The trap is sent by the SNMP agent to the set of trap manager's that are currently configured for the SNMP agent using the CHGSNMPA CL command. In other words, the subagent does not determine where the trap is sent.

This sequence is depicted in Figure 6-4.

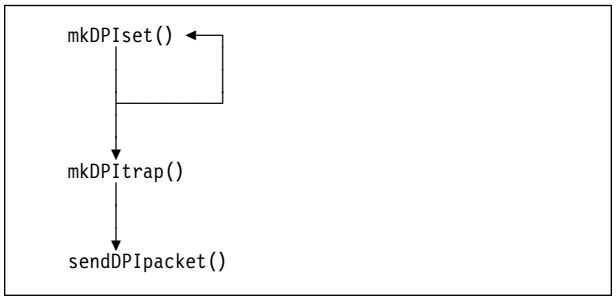


Figure 6-4. Subagent initiated trap.

ARE_YOU_THERE Request

A subagent can send an ARE_YOU_THERE packet to the agent. This may be useful to verify that the SNMP agent is still active and the connection is working.

If the connection is in a healthy state, the agent responds with a RESPONSE packet with SNMP_ERROR_DPI_noError. If the connection is not in a healthy state, the agent may respond with a RESPONSE packet with an error indication. But, the agent might not react at all. In this situation, you would time-out while waiting for a response.

Communicating with the SNMP Agent

An SNMP subagent communicates with the SNMP agent on the same AS/400 system using these APIs:

```
connectSNMP()
disconnectSNMP()
sendDPIpacket()
waitDPIpacket()
receiveDPIpacket()
```

The technical details about these APIs (parameters, return codes, and so on) can be found in "Simple Network Management Protocol (SNMP) Subagent APIs" the book *System API Reference: UNIX-Type APIs*. This section provides some additional information on how to use these APIs.

The connectSNMP() call is the very first subagent API that is used. The call establishes a logical connection with the SNMP agent (running on the same AS/400 as the subagent) and prepares the agent for the next logical subagent event. The next event is to perform a DPI open function (see mkDPIopen() section). The parameters to connectSNMP() specify a data queue that the subagent wants the SNMP agent to use when sending work to the subagent. This data queue name is used by the agent (For example, in the SNMP subagent MIB. The SNMP subagent MIB is described in "SNMP Subagent MIB" on page A-6.) as part of the identity for a subagent. Hence, only a single subagent may use a particular data queue at any one time. Note that this API does not create the data queue for the suba-

gent — the data queue should have already been created when this call is made.

The disconnectSNMP() call is the logical opposite of the connect function and is the very last subagent API that is used. The call ends or closes the logical connection between the SNMP agent and the subagent. All DPI subagent functions are preformed within the logical bracket formed by a connect and disconnect.

At any given time, a subagent may have:

- 0 or 1 connections with the SNMP agent
- 0 or 1 opens with the SNMP agent
- 0 or n registrations (no logical upper bound)

Zero opens is not a useful normal condition for a subagent. This would occur briefly only between a connectSNMP() call and the sending of a DPI open packet. Zero registrations might be useful for a subagent that only sends traps and does not implement any MIB groups. More than a single subtree registration may be useful for a variety of reasons:

1. Perhaps separate MIBs are being implemented by separate people, but it is decided to have these MIB implementations all run within a single job, which will perform the subagent API functions on behalf of the MIB implementers
2. Perhaps a few OIDs within a MIB are known to be relatively large overhead, compared to the rest of the subtree, for a DPI request type (for example, set). There may be performance benefits to registering these OIDs separately than the rest of the subtree, with a different time-out value.

The sendDPIpacket() API does just this — for any type of DPI packet a subagent wants to send, this routine is used to send it.

The waitDPIpacket() and receiveDPIpacket() are alternative ways of getting responses and work from the SNMP agent. Generally, a given subagent implementation chooses one or the other, and does not need both. But both can be used in the same subagent, if desired. The difference between the two APIs is that waitDPIpacket() completely handles the subagent's data queue, logically doing these steps:

1. Check data queue for an incoming message
2. When data queue has a message, receive the message
3. Check the message, if not from SNMP agent, return it to caller
4. If the message is from the SNMP agent, copy DPI packet to subagent buffer

In contrast, receiveDPIpacket() does only the last step. In which case, the subagent implementation must perform the first three. (Note that the DPI packet itself is not the message that is placed on the subagent's data queue. See the sa_dataq_msg structure in qtossapi.h for the format of the data queue message from the SNMP agent. The purpose of the data queue message, is to signal the subagent that a DPI packet is pending.)

Waiting for Work from the SNMP Agent

The main purpose for which an SNMP subagent is developed is to provide additional MIB groups to SNMP manager applications. Therefore, an SNMP subagent spends most of its time waiting for (and processing) requests that an SNMP manager has sent to the local SNMP agent.

Figure 6-5 on page 6-8 shows the structure for the core processing loop of a subagent implementation.

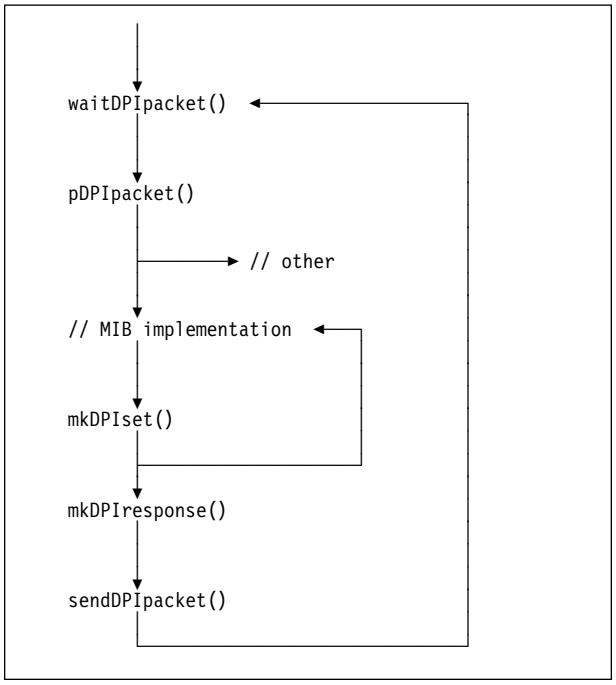


Figure 6-5. Normal processing loop for a subagent.

The waitDPIpacket() is at the top of the loop. When a request arrives from the SNMP agent, this API will receive it, verify that it is from the SNMP agent, and then return it to the caller. The DPI packet is then parsed by using the pDPIpacket() routine so that the packet contents are available.

A key decision point occurs after a successful parse. The DPI packet may be a GET, GETNEXT, or SET, in which case the path to the MIB implementation is taken. If it is some other DPI packet (for example, unregister), the subagent will take some other appropriate action.

The MIB implementation is the heart of a subagent and generally comprises well over one half of the total subagent implementation size. (Of course, it can go much higher if the MIB being implemented is very large.) The MIB code performs the requested operation (GET, GETNEXT, SET, COMMIT OR UNDO) on an OID, and a resulting varBind is built using the mkDPIset() routine.

If the incoming DPI packet had multiple varBinds, it can be handled as the loop suggests; by going back into the MIB implementation for the next OID, and adding that result to the varBind list using mkDPIset().

When all the varBinds have been processed, the list of varBinds is used to build a response DPI packet by calling mkDPIresponse().

Lastly, the DPI response is sent back to the waiting SNMP agent by calling sendDPIpacket(). Then the subagent calls waitDPIpacket() again, with an appropriate time-out, to again another cycle of the normal processing loop.

Part 3. Appendixes

Appendix A. OS/400 SNMP Agent Set Processing and Supported SNMP MIBs

This appendix lists and describes the following MIBs that are related to SNMP:

Standard RFC MIBs

- MIB II (RFC1213)
- Ethernet-like (RFC1398)
- FDDI (RFC1285)
- Frame Relay (RFC1315)
- Token Ring (RFC1231)

IBM enterprise MIBs

- APPN MIB
- Client management MIB
- NetView for AIX subagent MIB
- DPI 2.0 (RFC1592)
- SNMP subagent MIB

For more detailed information on these and other MIB groups, please refer to the concise MIB description.

MIBs are distributed through the following mechanisms:

- If you have Internet access with FTP capability, MIB modules can be obtained from the NetView Association MIB server at `netview.cary.ibm.com`.
- IBM enterprise MIB modules supported by OS/400 are shipped with OS/400. Each MIB module is contained in a member of physical file QANMMIB in library QSYS. The physical file member name for each MIB module is listed as the MIB module name in the section that describes each enterprise MIB in this appendix.
- If you have Internet access with FTP capability, MIB modules for vendor MIBs registered under the enterprises subtree (including IBM MIBs) can be obtained by anonymous FTP from the Internet Assigned Numbers Authority (IANA) anonymous FTP server at `venera.isi.edu`.
- As a last resort, MIB distribution can be initiated through the Internet address: `as400_sysman@vnet.ibm.com`

OS/400 SNMP Agent Set Processing

RFC 1157 requires that each set request variable assignment be effected as if simultaneously set with respect to all other assignments specified in the same message. This means that a set request with multiple instance ID/value pairs should be processed in an all-or-none fashion. That is, either all the new values of the variables are assigned without error, or else none of the values of the variables in the request are changed. This requirement is also known as atomic commit with rollback.

For the MIB objects that are supported, the OS/400 SNMP agent checks the specified values for the MIB variables in the set request. If a specified value does not meet the check requirements, the set request is rejected. The actual implementation for the set request is technically a best effort, not a true atomic commit and rollback.

During the set process when the agent is actually changing MIB object values, if a failure occurs, the original values of the MIB objects already set are not restored.

Standard RFC MIBs

MIB-II

Overview: MIB-II describes those objects that are implemented by managed nodes that run the Internet suite of protocols.

RFC: RFC 1213

RFC noted exceptions: The egp group is not supported. The set operation is not supported for the following MIB object, which is defined as having read-write access.

ifAdminStatus

Note that ifAdminStatus tracks the desired state of a network interface as set with an OS/400

command (for example, NETSTAT). Therefore, the get operation can still be used on ifAdminStatus and ifOperStatus to determine if there is a problem with an interface.

The set operation is accepted for the following MIB objects, which are defined as having read-write access. However, the values will not change as a result of the set operation. This behavior allows an SNMP manager to successfully perform a set operation on an entire row of a table. A subsequent get operation will show which values have actually changed.

- ipRouteIfIndex
- ipRouteMetric1
- ipRouteMetric2
- ipRouteMetric3
- ipRouteMetric4
- ipRouteAge
- ipRouteMetric5

For some MIB objects, the set operation is not supported for all values that are defined as being valid. These MIB objects are listed here, along with the values to which they can be set.

ipRouteType	invalid(2), indirect(4)
ipNetToMediaType	invalid(2), static(4)
tcpConnState	deleteTCB(12)

In order to set the value of atNetAddress, its syntax must be encoded as OCTET STRING, rather than NetworkAddress. Note that it is never necessary to set the value of atNetAddress directly. This is because a row can be added to or deleted from the atTable by setting the value of only atPhysAddress.

The result of changing the value of a MIB object that is an index to an instance of that MIB object is undefined. For example, the result of the operation `set ipRouteDest.9.130.38.28=9.130.38.29` is undefined.

When adding a row to a table by setting the value of a MIB object, default values are assigned to the other objects in the row.

- *indexes*: The value of a MIB object which is an index to an instance of that MIB object need not be set explicitly. For example, the operation `set ipRouteNextHop.9.130.38.28=9.130.25.250` will implicitly create the MIB object instance ipRouteDest.9.130.38.28, with the value 9.130.38.28.
- *atPhysAddress*: There is no default value for this MIB object. The value of this MIB object must be set in order to create a new row in the atTable.
- *ipRouteNextHop* There is no default value for this MIB object. The value of this MIB object must be set in order to create a new row in the ipRouteTable.
- *ipRouteType*: indirect(4)
- *ipRouteMask*: Corresponds to the network class. For example, the mask of a class-A network will default to 255.0.0.0.
- *ipNetToMediaPhysAddress*: There is no default value for this MIB object. The value of this MIB object must be set in order to create a new row in the ipNetToMediaTable.
- *ipNetToMediaType*: static(4)

MIB Subtree Description: Management information base for network management of TCP/IP-based internets.

MIB Subtree Object Identifier: mib-2 ::= { mgmt (1) }

Prerequisite MIB Modules: RFC1155, RFC1212

Ethernet-like Interface MIB

Overview: Ethernet-like Interface MIB defines objects for managing ethernet-like object interface types.

RFC: RFC1398

RFC noted exception: The dot3CollTable is not supported.

MIB Subtree Description: Ethernet-like MIB.

MIB Subtree Object Identifier: dot3 ::= { transmission 7 }

Prerequisite MIB Modules: RFC1155, RFC1213, RFC1212

FDDI MIB

Overview: FDDI MIB defines objects for managing devices which implement the FDDI.

RFC: RFC1285

RFC noted exceptions: The Attachment group is not supported. The set operation is not supported for this MIB.

MIB Subtree Description: FDDI MIB.

MIB Subtree Object Identifier: fddi ::= { transmission 15 }

Prerequisite MIB Modules: RFC1155, RFC1213, RFC1212

Frame Relay MIB

Overview: Frame Relay objects for managing Frame Relay.

RFC: RFC1315

RFC noted exception: The set operation is supported for only frTrapState.

MIB Subtree Description: Frame Relay MIB.

MIB Subtree Object Identifier: frame-relay ::= { transmission 32 }

Prerequisite MIB Modules: RFC1155, RFC1213, RFC1212, RFC1215

IEEE 802.5 Token Ring MIB

Overview: IEEE 802.5 Token Ring MIB defines managed objects used for managing sub-networks which use the IEEE 802.5 Token Ring technology described in 802.5 Token Ring Access Method and Physical Layer Specifications, IEEE Standard 802.5-1989.

RFC: RFC1231

RFC noted exception: The dot5TimerTable and the SET operation are not supported for this MIB.

MIB Subtree Description: Token Ring MIB

MIB Subtree Object Identifier: dot5 ::= { transmission 9 }

Prerequisite MIB Modules: RFC1155, RFC1212

IBM Enterprise MIBs

Advanced Peer-To-Peer Networking (APPN) MIB

Overview: The APPN Node Group provides global information about the APPN node, which is either a network node or an end node.

The APPN Topology Group represents the entire APPN network topology including network nodes, virtual nodes, and all transmission groups (TGs) that are associated with these nodes.

The APPN Local Topology Group describes the local topology. This MIB group defines the required objects for retrieval of information about this node and the objects that represent the local topology about end nodes.

The APPN port table provides information on APPN ports and allows their state to be changed from an SNMP manager using SNMP SET on the port state object. (On the AS/400, APPN ports are APPN-capable lines.)

The APPN link station table provides information on APPN link stations and allows their state to be changed from an SNMP manager using SNMP SET on the link station state object. (On the

AS/400, APPN link stations are APPC controllers that are attached to APPN-capable lines.)

RFC: Informational RFC 1593

RFC noted exceptions: Some groups of RFC 1593 are implemented with some extensions. Refer to the concise MIB description (MIB module) for details.

MIB Subtree Description: SNA APPN MIB

MIB Subtree Object Identifier: ibmappn ::= { internet(1) private(4) enterprises(1) ibm(2) prod(6) ibm6611(2) 13 }

MIB Module Name: IBMAPPN

The ASN.1 for this MIB is available in member IBMAPPN in file QSYS/QANMMIB.

Prerequisite MIB Modules: RFC1155, RFC1212

APPN MIB object groups that supported: Groups that are supported under the ibmappnNode subtree:

- ibmappnGeneralInfoAndCaps
- ibmappnNnUniqueInfoAndCaps
- ibmappnEnUniqueCaps
- ibmappnSnmpInformation

Groups that are supported under the ibmappnNn subtree:

- ibmappnNnTopo (only 4 objects are supported):
  - ibmappnNnTopoMaxNodes
  - ibmappnNnTopoCurNumNodes
  - ibmappnNnTopoNodePurges
  - ibmappnNnTopoTgPurges
- ibmappnNnTopology - tables for APPN network topology:
  - ibmappnNnTopologyTable
  - ibmappnNnTgTopologyTable
  - ibmappnNnTopologyFRTable
  - ibmappnNnTgTopologyFRTable

Groups that are supported under the ibmappnLocalTopology subtree:

- ibmappnLocalThisNode
  - ibmappnLocalGeneral
  - ibmappnLocalNnSpecific
  - ibmappnLocalTg - table of local transmission groups (TGs)
  - ibmappnLocalEnTopology
- ibmappnLocalEnTable - table of adjacent end nodes
- ibmappnLocalEnTgTable - table of adjacent end node TGs

The following objects are supported from the APPN port table:

- ibmappnNodePortName
- ibmappnNodePortState
- ibmappnNodePortDlcType
- ibmappnNodePortPortType
- ibmappnNodePortLsRole
- ibmappnNodePortMaxRcvBtuSize

The list of supported link station table objects follows. Any AS/400-specific considerations regarding the objects are listed in parenthesis after each object name.

- ibmappnNodeLsName
- ibmappnNodeLsPortName
- ibmappnNodeLsDlcType
- ibmappnNodeLsDynamic (controller value for control owner)
- ibmappnNodeLsState
- ibmappnNodeLsCpName
- ibmappnNodeLsTgNum
- ibmappnNodeLsLimResource (controller value for switched disconnect)
- ibmappnNodeLsBlockNum
- ibmappnNodeLsIdNum
- ibmappnNodeLsCpCpSession
- ibmappnNodeLsEffCap (0 implies *MAX, 1 implies *MIN)
- ibmappnNodeLsConnCost
- ibmappnNodeLsByteCost
- ibmappnNodeLsSecurity
- ibmappnNodeLsDelay
- ibmappnNodeLsUsr1
- ibmappnNodeLsUsr2
- ibmappnNodeLsUsr3
- ibmappnNodeLsHprSup: yes(1) iff APPN HPR capable is *YES
- ibmappnNodeLsErrRecoSup

Client Management MIB

Overview: The Client System Group provides global information about the client, connectivity, and capabilities.

The Client Hardware Group provides information that is related to storage, devices, file systems, and printers.

The Client Software Group provides information that is related to installed software and fixes.

RFC: None

RFC noted exception: None

MIB Subtree Description: Client Management

MIB Subtree Object Identifier:

clientMgmtSubAgent ::= { internet(1) private(4) enterprise(1) ibm(2) ibmprod(6) 50 }

MIB Module Name: IBMCLTM

The ASN.1 for this MIB is available in member IBMCLTM in file QSYS/QANMMIB.

Prerequisite MIB Modules: RFC1155, RFC1212

NetView for AIX Subagent MIB

Overview: One MIB object in this MIB is supported. The object provides the average percentage of load (processor utilization) during the elapsed time. Each retrieval of this value is calculated in the same manner as the value that is displayed by the DSPSYSACT command when the *restart* function key is used.

RFC: None

RFC noted exception: None

MIB Subtree Description: NetView for AIX subagent ComputerSystem group

MIB Subtree Object Identifier:

nv6saComputerSystem ::= { internet(1) private(4) enterprises(1) ibm(2) prod(6) netView6000SubAgent(4) 5 }

MIB Module Name: IBMNV6SA

The ASN.1 for this MIB is available in member IBMNV6SA in file QSYS/QANMMIB.

Prerequisite MIB Modules: RFC1155, RFC1212

Note: A NetView for AIX application provides support for this data.

NetView for AIX subagent MIB object supported

nv6saComputerSystemLoad OBJECT-TYPE

SYNTAX Gauge
ACCESS read-only
STATUS mandatory
DESCRIPTION
The CPU load as a percentage. For example, 25% is 2500.
::= { nv6saComputerSystem 1 }

DPI 2.0 MIB

Overview: An extension to SNMP agents that permits end-users to dynamically add or replace management variables in the local MIB without requiring recompilation of the SNMP agent. Although SNMP subagents are supported by an AS/400 system, this support does not use the DPI 2.0 MIB. This is because the AS/400 subagent support does not use TCP or UDP between the agent and subagents. A transparent, internal mechanism is used by the subagents. This MIB is supported by the AS/400 SNMP agent even though the communication between the agent and subagent on the AS/400 does not use TCP or UDP. This is done for compatibility.

This MIB is implemented by the AS/400 system but is not currently used. For detail on the MIB, see RFC1592.

RFC: RFC1592

RFC noted exception: None.

MIB Subtree Description: SNMP Distributed Protocol Interface Version 2.0

MIB Subtree Object Identifier: dpi20MIB ::= { ibmDPI 1 }

Prerequisite MIB Modules: SNMPv2-SMI

SNMP Subagent MIB

Note!

This MIB is experimental and is subject to change in future releases.

Overview: The SNMP subagent MIB provides information about subagents to facilitate management activities. The MIB is designed to handle multiple types of subagents, including DPI subagents (see RFC1592), for which the DPI 2.0 API is provided. All the SA MIB information is dynamic, for the duration of the SNMP agent job. The SA MIB consists of six summary OIDs and two tables. The summary OIDs are:

saDefaultTimeout
default value for subagents for response timeout

saMaxTimeout
largest value a subagent may use for response time-out

saAllowDuplicateIDs
flag to allow duplicate subagent OIDs, or not

saNumber
number of currently registered subagents

saAllPacketsIn
total number of subagent packets that are received from all subagents

saAllPacketsOut
total number of subagent packets that are sent to all subagents.

The saTable provides information about specific subagents such as status, address, and description.

The saTreeTable provides information about specific subtrees that subagents have registered with the SNMP agent.

RFC: None

RFC noted exception: None

MIB Subtree Description: SNMP subagent MIB

MIB Subtree Object Identifier: saMIB ::= { internet(1) private(4) enterprise(1) ibm(2) ibmResearch(2) 12 }

MIB Module Name: IBMSNMPSA

The ASN.1 for this MIB is available in member IBMSNMPSA in file QSYS/QANMMIB.

Prerequisite MIB Modules: RFC1155, RFC1212, RFC1213

Appendix B. Journal for SNMP Logging

This appendix lists the format of journal entries for SNMP logging. It also provides examples of different display journal entries. SNMP agent logging records are kept in journal QSNMP in library QUSRSYS. Each journal record contains 118 columns of alphanumeric characters. For more information on the use of journals, see the *Backup and Recovery* book.

The format of journal entries for SNMP agent logging is listed below.

```
column 1:  log type
              - G:  GET request
              - N:  GETNEXT request
              - S:  SET request
              - R:  response
              - T:  trap
    2-21:  community name
   22-33:  internet address of SNMP manager sending request or receiving
             response, or internet address of first SNMP trap manager
             receiving trap
      34:  error status
              - 0:  noError
              - 1:  tooBig
              - 2:  noSuchName
              - 3:  badValue
              - 5:  genErr
   35-38:  error index
      39:  trap type
              - 0:  coldStart
              - 1:  warmStart
              - 2:  linkDown
              - 3:  linkUp
              - 4:  authenticationFailure
              - 5:  egpNeighborLoss
              - 6:  enterpriseSpecific
   40-43:  enterprise specific trap type
           NOTE: If the enterprise specific trap type value is less
                 than -999, it will be logged as -999.  If the
                 enterprise specific trap type value is greater than
                 9999, it will be logged as 9999.
  44-118:  object descriptors
```

The following examples of journal entries are displayed when using the CL command *DSPJRN JRN(QUSRSYS/QSNMP)*.

- This entry indicates that a **GET** request specifying community private was sent by an SNMP manager at internet address 9.130.38.28. The error status, error index, and trap type fields all have the value zero. The object descriptors that are specified in the **GET** request are shown.

```
- Column      *...+....1....+....2....+....3....+....4....+....5
  00001      'Gprivate              9130 38 280   00   0sysUpTi'
  00051      'me.0 sysUpTime                                    '
  00101      '                  '
```

- This entry is the log of the response to the previous GET request. The error status fields and error index fields show that the noSuchName error was returned for the second object that is specified in the GET request.

```
- Column      *...+....1....+....2....+....3....+....4....+....5
  00001      'Rprivate              9130 38 282   20   0sysUpTi'
  00051      'me.0 sysUpTime                                    '
  00101      '                  '
```

- This entry indicates that a GETNEXT request specifying community private was sent by an SNMP manager at internet address 9.130.38.28. Error status, error index, and the trap type fields all have the value zero. The object descriptors that are specified in the GETNEXT request are shown.

```
- Column      *...+....1....+....2....+....3....+....4....+....5
  00001      'Nprivate              9130 38 280   00   0at atIf'
  00051      'Index.1.1.9.130.25.58 atIfIndex.1.1.9.130.25.195  '
  00101      '                  '
```

- This entry is the log of the response to the previous GETNEXT request. Other than the log type field, only the object descriptors field has changed. It now shows the object descriptors of the objects that are returned in response to the GETNEXT request. Because the object descriptor of the third returned object cannot completely fit within the remaining object descriptors field space, the descriptor is not shown at all.

```
- Column      *...+....1....+....2....+....3....+....4....+....5
  00001      'Rprivate              9130 38 280   00   0atIfInd'
  00051      'ex.1.1.9.130.25.58 atIfIndex.1.1.9.130.25.195     '
  00101      '                  '
```

- This entry indicates that a trap was sent to the trap manager at internet address 9.130.38.154. Error status and error index are both zero, and the trap type is linkUp. The link that has come online is associated with the object descriptor ifIndex.3. The trap did not specify a community name.

```
- Column      *...+....1....+....2....+....3....+....4....+....5
  00001      'T                     9130 381540   03   0ifIndex'
  00051      '.3                                                '
  00101      '                  '
```

Appendix C. Problem Analysis for SNMP

This appendix is to be used for determining solutions to problems that are encountered while using SNMP.

Problem Analysis for SNMP Agent

The most common problems, causes, and solutions for the SNMP agent are shown in Table C-1.

Table C-1 (Page 1 of 3). SNMP Agent Problem Analysis

Problem	Cause	Solution
SNMP agent jobs do not start when the STRTCP command runs.	SNMP attribute AUTOSTART is set to *NO.	Use the CHGSNMPA command to change AUTOSTART to *YES.
SNMP agent jobs do not start when the STRTCPSVR command runs.	Another application is listening on UDP port 161.	Make sure no other applications are using UDP port 161 and then start the SNMP agent again using the STRTCPSVR command.
SNMP managers are not receiving any responses from the OS/400 SNMP agent for any SET, GET, or GETNEXT requests.	The OS/400 SNMP agent is not active.	Make sure TCP/IP (or AnyNet) and the OS/400 SNMP agent are active. The OS/400 SNMP agent is active if jobs QTMSNMP and QTMSNMPRCV are running in subsystem QSYSWRK. Use the WRKACTJOB command to determine if these jobs are active.
	The SNMP manager is sending the request to a system other than the intended AS/400.	Make sure that the SNMP manager is sending the request to the intended AS/400.
	The SNMP manager is comparing the response PDU source IP address with the destination IP address in the PDU it sent, and they are not equal.	The SNMP agent will always send the response PDU to the IP address which sent the original PDU. Verify, especially for an SNMP manager system with multiple IP-addresses, that the manager is using the expected IP address.
	The SNMP manager is comparing the source UDP port number in the response PDU with the destination UDP port number (the well-known port 161) in the PDU it sent to OS/400, and they are not equal.	The OS/400 SNMP agent does not use UDP port 161 to send response PDUs, due to technical reasons. The SNMP management application should not check the response PDU port number. Instead, the SNMP manager application needs to rely on checking the response PDU IP address and, within the PDU, the request-id to verify the PDU.
	The SNMP manager is specifying a community name that is unknown to the OS/400 SNMP agent.	Make sure that the SNMP manager is specifying a community name that is known to the OS/400 SNMP agent. The list of community names may be displayed by using the CFGTCPSNMP command. Community names are case-sensitive, so make sure that the SNMP manager is specifying the community name correctly (example: PUBLIC and public are two different community names). Also, make sure that the value for the Translate community name (ASCIICOM) parameter for the OS/400 community corresponds to the community name being specified by the SNMP manager. If the SNMP manager is an ASCII system, the ASCIICOM parameter for the OS/400 community should be *YES. If the SNMP manager is an EBCDIC system or the community name has one or more characters that cannot be displayed, the ASCIICOM parameter for the OS/400 community should be *NO. If the community name or ASCIICOM parameter need to be changed, the community must be removed by using the RMVCOMSNMP command and then added with the correct values by using the ADDCOMSNMP command.

Table C-1 (Page 2 of 3). SNMP Agent Problem Analysis

Problem	Cause	Solution
	The SNMP manager is specifying a community name that is known to the OS/400 SNMP agent, but the IP address of the SNMP manager is not part of the community.	Make sure that the IP address of the SNMP manager is defined in the OS/400 community. If the system the manager is running on has multiple IP addresses, ensure that the IP address used to send UDP packets to the AS/400 agent is the included in the IP addresses for the community name configuration intended for the manager.
		The SNMP agent has the capability to selectively log (in a journal QSNMP, in library QUSRSYS) get and set PDUs and traps. When logging get and set PDUs, both the incoming and outgoing PDUs are logged. These are controlled with the CHGSNMPA and CHGCOMSNMP commands. Verify, with the SNMP logging capabilities, that the expected PDUs are being received and sent by the SNMP agent. See Appendix B, "Journal for SNMP Logging" on page B-1 for information about how to use this function. The IP address of the SNMP manager can be added to the community by using the CHGCOMSNMP command.
	Object access in the OS/400 community definition specifies *NONE. Object access in the OS/400 community definition specifies *SNMPATR and object access in the OS/400 SNMP attributes is *NONE.	Make sure object access for the community is either *READ or *WRITE depending on if you desire SET requests to be honored. The object access may be changed by using the CHGCOMSNMP command.
	There are problems with the TCP/IP network or AnyNet network support.	See the *TCP/IP Configuration and Reference* for more information about network problem analysis. If network problems are suspected, use the System Service Tool (STRSST) to generate and format a communications trace of UDP datagrams, to verify that expected PDUs are arriving and being sent by the AS/400.
SNMP managers are receiving error responses from the OS/400 SNMP agent for GET or GETNEXT requests.	The SNMP manager is specifying an incorrect object identifier in the request. The SNMP manager is specifying an object identifier for an object that is not supported by the OS/400 SNMP agent.	Make sure the SNMP manager is specifying an object identifier for an object that is supported by the OS/400 SNMP agent.
SNMP managers are receiving error responses from the OS/400 SNMP agent for SET requests.	The SNMP manager is specifying an incorrect object identifier in the request.	Make sure the SNMP manager is specifying an object identifier for an object that is supported by the OS/400 SNMP agent.
	The SNMP manager is specifying an object identifier for an object that is not supported by the OS/400 SNMP agent.	Make sure the SNMP manager is specifying an object identifier for an object that can be changed.
	The SNMP manager is attempting to set an object that is defined as read-only. The SNMP manager is attempting to set an object to a value that is not valid.	Make sure the SNMP manager is specifying a valid value for the object.
	The object access for the community is not *WRITE. The object access for the community is *SNMPATR and the object access in the OS/400 SNMP attributes is not *WRITE.	Make sure that the object access for the OS/400 community is *WRITE. The object access for the community may be changed by using the CHGCOMSNMP command. The object access in the SNMP attributes may be changed by using the CHGSNMPA command.
SNMP managers are unable to access any objects found in the APPN MIB, Client Management MIB, or NetView for AIX MIB.	The OS/400 subagent job is not active. The OS/400 subagent job is active, but is inactive according to the OS/400 SNMP agent.	End the SNMP agent by using the ENDTCPSVR command and then start the SNMP agent by using the STRTCPSVR command. The OS/400 subagent runs in job QSNMPSA in subsystem QSYSWRK. Use the WRKACTJOB command to determine if this job is active.

Table C-1 (Page 3 of 3). SNMP Agent Problem Analysis

Problem	Cause	Solution
Traps are not being received by an SNMP manager.	The IP address of the intended SNMP manager is not specified correctly in the OS/400 SNMP attributes.	Make sure that the IP address of the SNMP manager to receive the trap is specified correctly in the OS/400 SNMP attributes. The IP address of the SNMP manager may be changed by using the CHGSNMPA command.
	The community name to be placed in the trap is not recognized by the SNMP manager.	Make sure the trap community name is specified correctly in the OS/400 SNMP attributes. The trap community name may be changed by using the CHGSNMPA command.
		The SNMP agent has the capability to log traps (in journal QSNMP, in library QUSRSYS). This is controlled with the CHGSNMPA command. Verify, with the SNMP logging capabilities, that the expected trap PDUs are being sent by the SNMP agent. See Appendix B, "Journal for SNMP Logging" on page B-1 for information about how to use this function.
	There are problems with the TCP/IP network or AnyNet support.	See the *TCP/IP Configuration and Reference* for more information about network problem analysis.
authenticationFailure traps are not being received by an SNMP manager.	The IP address of the intended SNMP manager is not specified correctly in the OS/400 SNMP attributes.	Make sure that the IP address of the SNMP manager to receive the trap is specified correctly in the OS/400 SNMP attributes. The IP address of the SNMP manager may be changed by using the CHGSNMPA command.
	The community name to be placed in the trap is not recognized by the SNMP manager.	Make sure the trap community name is specified correctly in the OS/400 SNMP attributes. The trap community name may be changed by using the CHGSNMPA command.
	The Send authentication traps (SNDAUTTRP) SNMP attribute is specified as *NO.	Make sure the SNDAUTTRP SNMP attribute is specified as *YES. The SNDAUTTRP SNMP attribute may be changed by using the CHGSNMPA command.
		Verify, with the SNMP trap logging capabilities, that the expected trap PDUs are being sent by the SNMP agent. See Appendix B, "Journal for SNMP Logging" on page B-1 for information about how to use this function.
	There are problems with the TCP/IP network or AnyNet support.	See the *TCP/IP Configuration and Reference* for more information about network problem analysis.
Entries are not being made in journal QSNMP in library QUSRSYS for SET requests received from SNMP managers.	SET logging is specified as *NO in the OS/400 community definition. SET logging is specified as *SNMPATR in the OS/400 community definition and is specified as *NO in the OS/400 SNMP attributes.	Make sure that SET logging is specified as *YES in the OS/400 SNMP community definition. The SET logging for the community may be changed by using the CHGCOMSNMP command. The SET logging in the SNMP attributes may be changed by using the CHGSNMPA command.
Entries are not being made in journal QSNMP in library QUSRSYS for GET or GETNEXT requests received from SNMP managers.	GET logging is specified as *NO in the OS/400 community definition. GET logging is specified as *SNMPATR in the OS/400 community definition and is specified as *NO in the OS/400 SNMP attributes.	Make sure that GET logging is specified as *YES in the OS/400 SNMP community definition. The GET logging for the community may be changed by using the CHGCOMSNMP command. The GET logging in the SNMP attributes may be changed by using the CHGSNMPA command.
Entries are not being made in journal QSNMP in library QUSRSYS for traps sent to SNMP managers.	Trap logging is specified as *NO in the OS/400 SNMP attributes.	Make sure that trap logging is specified as *YES in the OS/400 SNMP attributes. The trap logging in the SNMP attributes may be changed by using the CHGSNMPA command.

Problem Analysis for SNMP Manager APIs

The most common problems, causes, and solutions for the SNMP Management application developer are shown in Table C-2.

Table C-2 (Page 1 of 2). SNMP Management application developer problem analysis

Problem	Cause	Solution
The SNMP management application received an out of memory or out of buffers error.	The SNMP API has run out of resources to complete the operation.	The management application can try reducing the size of the requested operation. If the problem persists, report the error using the ANZPRB command.
The SNMP Management application received an out of varBinds error.	The requested operation has exceeded the allowable number of varBinds for a single operation.	The management application can try reducing the number of varBinds in the varBind list.
The SNMP Management application received an invalid OID error.	An Object IDentifier in the varBind list was not specified in the correct dotted decimal notation.	The management application should specify the Object IDentifier in the correct dotted decimal notation.
The SNMP Management application receives an invalid value error.	The SNMP API's will do some rudimentary checking on the value supplied during a snmpSet operation. The API's will detect an incorrect length on integer types, and will also check for incorrect dotted decimal notation on IP addresses and OID's when supplied as values.	The management application should specify the value on a set in the correct form as specified by the ASN type.
The SNMP Management application receives an invalid value representation error.	The ASN type specified for a particular OID in the varBind list is not recognized as a common ASN type.	The management application should specify a known ASN type as listed in the QTOMEAPI CLEINC file in QSYSINC library. (If you cannot locate this file, contact your system support department. The QSYSINC library can be selectively installed at any time.)
The SNMP Management application receives an encode or decode error.	The PDU specified could not be encoded or decoded for transmission across the wire.	Retry the command. If the problem persists, report the error to IBM using the ANZPRB command.
The SNMP Management application received an invalid community name length error.	The value in community name length field was not in the range of 1 to 256.	The management application should specify a community name length that is greater than 0 and less than or equal to 256.
The SNMP Management application received a time-out parameter error.	The value in the time-out parameter field was not in the range of 1 to 100.	The management application should specify a time-out value that is greater than 0 and less than or equal to 100.
The SNMP Management application received an unknown host error.	The hostname specified was not recognized as a valid host on the network.	The management application should specify the correct hostname, or specify the dotted decimal notation of the IP address.
The SNMP Management application received a not OK error.	The value length field on a varBind in the varBind list was less than 0.	The management application should specify a value field greater than or equal to 0.
SNMP management application receives a time-out from the SNMP APIs.	The time-out value is set too low.	Increase the time-out value and try again.
	The SNMP agent receiving the SNMP operation is not active.	Make sure that the receiving agent is active.
	The SNMP management application is specifying a community name that is unknown to the SNMP agent.	Specify the correct community name.
	The SNMP management application is specifying a correct community name, but the IP address of the manager is not part of the community.	Make sure the IP address is know to the SNMP agent as a valid SNMP manager.
	There are problems with the TCP/IP network or AnyNet support.	See the *TCP/IP Configuration and Reference* for more information about network problem analysis.

Table C-2 (Page 2 of 2). SNMP Management application developer problem analysis

Problem	Cause	Solution
SNMP management application receives a sockets error while trying to initiate a SNMP operation.	The SNMP API could not connect properly to sockets to perform its operation across TCP/IP or AnyNet.	Make sure TCP/IP support or AnyNet support is active on the system, and then retry the command.
SNMP management application receives an invalid PDU type.	The SNMP management application tried to issue an SNMP operation of one type while specifying a PDU built for another type.	Make sure that the PDU type matches that of the SNMP operation being performed.
SNMP management application receives an invalid IP address.	The SNMP management application tried to issue an SNMP operation with an unrecognizable IP address.	Make sure that the IP address is either in the Hostname form or in the dotted decimal form. (example of hostname form: host.city.state) (example of dotted decimal form: 9.9.9.9)
SNMP management application received a Domain Error, Invalid pointer, or Invalid pointer type return code from the SNMP APIs.	The SNMP management application specified a pointer which caused an object domain error, reference location in a space that does not contain a pointer, or the pointer referenced storage in system state. These errors are equivalent to escape messages MCH6801, MCH3601, and MCH3602.	The SNMP management application should not use system state references, retry the operation passing in a valid pointer to user state storage. See the previously discussed message descriptions for further information.
SNMP management application received an Invalid PDU, Host, Community, or Invalid pointer type return code from the SNMP APIs.	The SNMP management application specified a Null pointer.	The SNMP management application should use a non-NULL pointer.
SNMP management application received a Return code 1 from the APIs.	This states that the amount of space allocated for the value returned in one or more varBinds, for a GET or GETNEXT operation, was not sufficient.	The SNMP management application should specify a greater value in the val_len field of the _varBind structure.
SNMP management application receives a non-zero return status in the Error Status field of the SNMP PDU on a Get or GetNext operation.	The SNMP agent could not fit the contents of the returned data in the SNMP message.	If the management application specified more than one object identifier to retrieve, then reduce the number of object identifiers until the returned data fits into a SNMP message.
	The SNMP management application is specifying an incorrect object identifier in the request.	Make sure that the SNMP management application is specifying an object identifier for an object that is supported by the receiving SNMP agent or its subagents.
	The SNMP management application is specifying an object identifier that is not supported by the receiving SNMP agent or its subagents.	
SNMP management application receives a non-zero return status in the Error Status field of the SNMP PDU on a Set operation.	The SNMP management application is specifying an incorrect object identifier in the request.	Make sure the SNMP manager is specifying an object identifier for an object that is supported by the receiving SNMP agent or its subagents.
	The SNMP management application is specifying an object identifier that is not supported by the receiving SNMP agent or its subagents.	Make sure that the SNMP management application is specifying an object identifier for an object that can be changed.
	The SNMP management application is attempting to set and object that is defined as read-only.	
	The SNMP management application is attempting to set an object value that is not valid. This value could be the incorrect ASN syntax, or it may not be in the acceptable range of values.	Make sure that the SNMP management application is specifying the correct ASN syntax in the ASN type field. Also, Make sure that the SNMP management application is specifying the value with the acceptable range.

Problem Analysis for SNMP Trap Manager

The most common problems, causes, and solutions for the SNMP trap manager are shown in Table C-3 on page C-6.

Table C-3. SNMP Trap Manager Problem Analysis

Problem	Cause	Solution
SNMP trap manager jobs do not start when the STRTRPMGR command runs.	Another application is listening on UDP port 162.	Make sure no other applications are using UDP port 162 and then start the SNMP trap manager again using the STRTRPMGR command.
Traps are not being received by the SNMP trap manager.	The IP address of this system is not specified correctly on the SNMP agent system sending the trap.	Make sure that the IP address of this system is specified correctly on the SNMP agent system.
	There are problems with the TCP/IP network or AnyNet support.	See the *TCP/IP Configuration and Reference* for more information about network problem analysis.
Traps are not being forwarded to other SNMP manager systems.	The IP address of the intended SNMP manager is not specified correctly in the OS/400 SNMP attributes.	Make sure that the IP address of the SNMP manager to receive the trap is specified correctly in the OS/400 SNMP attributes. The IP address of the SNMP manager may be changed by using the CHGSNMPA command.
	The community name to be placed in the trap is not recognized by the SNMP manager.	Make sure the trap community name is specified correctly in the OS/400 SNMP attributes. The trap community name may be changed by using the CHGSNMPA command.
	Trap forwarding was not specified when the SNMP trap manager was started on this system.	Make sure that trap forwarding is specified when the SNMP trap manager is started. The SNMP trap manager is started by using the STRTRPMGR command.
	The SNMP agent on this system is not active.	Make sure the SNMP agent is active. The OS/400 SNMP agent is active if jobs QTMSNMP and QTMSNMPRCV are running in subsystem QSYSWRK. Use the WRKACTJOB command to determine if these jobs are active.
	There are problems with the TCP/IP network or AnyNet support.	See the *TCP/IP Configuration and Reference* for more information about network problem analysis.
Jobs QTMSNMP, QTMSNMPRCV, QTRAPMGR, and QTRAPMGRRCV in subsystem QSYSWRK are consuming a large amount of processor resource.	It is possible that this may be caused by trap forwarding being specified and the IP address of this system being listed as a trap manager in the OS/400 SNMP attributes. It is possible that this system is part of a ring of trap managers in your network that forward traps to each other.	Do one or more of the following: • Turn off trap forwarding by ending the trap manager using the ENDTRPMGR command. Then start the trap manager using the STRTRPMGR command with trap forwarding specified as *NO. • Remove the IP address of this system from the trap manager list in the OS/400 SNMP attributes using the CHGSNMPA command. • Make sure there are no rings of trap managers in your network.

Problem Analysis for SNMP Subagent APIs

The most common problems, causes, and solutions for the SNMP subagent developer are shown in Table C-4.

Table C-4 (Page 1 of 3). SNMP subagent developer problem analysis

Problem	Cause	Solution
A SNMP subagent cannot 'connect' to the SNMP agent.	SNMP agent job is not running.	Make sure the SNMP agent job on the AS/400 (QTCP/QTMSNMP) is running. (Use the WRKACTJOB CL command. Another way to check is to send a SNMP 'get' request for some easy OID (for example, sysContact.0) using a SNMP manager or a utility such as snmpinfo (widely available on Unix systems).)

Table C-4 (Page 2 of 3). SNMP subagent developer problem analysis

Problem	Cause	Solution
	A return code of snmpsa_RC_timedout was returned by connectSNMP().	Make sure that the duration of the time-out (the third parameter) is not too small, with respect to system load. Generally, 20 or so should be sufficient.
	A return code of snmpsa_RC_parmerr was returned by connectSNMP().	Make sure the data queue and library name parameters are not NULL pointers and are between 1 and 10 bytes in length, and specify valid library and queue names.
		Verify that the library name is not QTEMP.
The SNMP subagent is unable to successfully perform the open function	The subagent receives a NULL pointer from the mkDPIopen() routine.	A NULL pointer from mkDPIopen() is usually caused by a simple error in one of the parameters. Verify that each of the parameters in the call are correct.
		Verify that the subagent OID does not violate SNMPv2 limits; no more than 128 subids and each subid must be representable in a 32-bit unsigned integer field.
	The subagent cannot send the DPI open packet.	Verify that prior to calling sendDPIpacket(), a call has successfully been made to the connectSNMP() routine.
	The subagent receives an SNMP_ERROR_DPI_otherError (101) in the DPI response packet from the agent.	An SNMP_ERROR_DPI_otherError (101) in the DPI response packet from the agent usually occurs due to some invalid value in the DPI open packet. For example, the subagent's OID must be null terminated and end with a subid, not a '.'.
	The subagent receives an SNMP_ERROR_DPI_duplicate-SubagentIdentifier (109) in the DPI response packet from the agent.	The subagent receives a SNMP_ERROR_DPI_duplicate-SubagentIdentifier (109) when the SNMP agent already has a active subagent with the same OID, and the subagent MIB saAllowDuplicateIDs is set to 2. Either change the OID the subagent uses in the mkDPIopen() call, or change the subagent MIB saAllowDuplicateIDs to 1 to allow duplicates.
The SNMP subagent is unable to successfully perform the register function.	The subagent receives a NULL pointer from the mkDPIregister() routine.	A NULL return usually indicates some parameter error. Check the subagent's job log for exceptions. If any are found, correct them, rebuild the subagent code and try the call again. If no exceptions are found, verify that the parameters are valid (for example, group_p is not NULL).
	The subagent receives an SNMP_ERROR_DPI_otherError (101) in the DPI response packet from the agent.	The most common cause of an SNMP_ERROR_DPI_otherError (101) in response to a DPI register packet is that the subtree OID did not end with a '.'; verify that the subtree OID ends with a '.'.
	The subagent receives an SNMP_ERROR_DPI_alreadyRegistered (103) in the DPI response packet from the agent.	The most common cause of an SNMP_ERROR_DPI_alreadyRegistered (103) error is that the subagent attempted to register a subtree that would cause a protected subtree to be affected. Verify that the subtree to be registered (parameter group_p in the mkDPIregister() call) is not above, at, or within one of the SNMP agent's protected subtrees (see the list of these in the mkDPIregister() API documentation).
	The subagent receives an SNMP_ERROR_DPI_higher-priorityRegistered (104) in the DPI response packet from the agent.	For an SNMP_ERROR_DPI_higher-priorityRegistered (104), re-request the subtree registration with a higher priority or 0 (which requests the highest). If the already registered subagent has priority 0 (get the subagent MIB saTstatus for the subtree), then there is no higher priority to request, so the other subtree must be unregistered or its subagent end before you can register the subtree.

Table C-4 (Page 3 of 3). SNMP subagent developer problem analysis

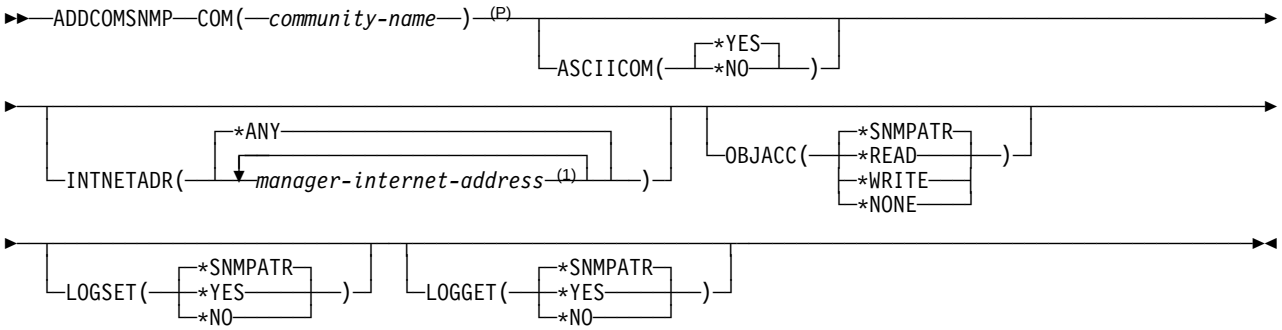
Problem	Cause	Solution
The SNMP subagent has successfully done connect, open and register, but does not receive any DPI packets (from the SNMP agent).	The SNMP agent job is not running.	Make sure the SNMP agent job on the AS/400 (QTCP/QTMSNMP) is running (use the WRKACTJOB CL command) and responding to PDUs in a normal way.
	There have not been any SNMP PDUs for the subtrees registered by the subagent.	Verify that a SNMP PDU (Get, Getnext or Set) for an OID in the subagent's subtree, has been sent to the SNMP agent.
	The subagent or the subagent's subtree registration status is not 'valid' (1).	Check the status OIDs in the subagent MIB and verify the values are 'valid' (1). These OIDs are in the SNMP subagent MIB in the saTable and saTreeTable, and are the saStatus OID (for the right subagent) and saTstatus OID (for the right subtree). (The ASN.1 definition for the subagent MIB can be found in QSYS/QANMMIB, member IBMSNMPSA.)
The SNMP subagent works for awhile, but then mkDPIset() return NULL, when trying to build structure for a response packet.	There is insufficient memory to build the internal structures for the DPI packet, because it has not been freed, after having been dynamically allocated.	Ensure that the fDPIset() routine is called so that memory use does not monotonically increase while the subagent is running. (see "Simple Network Management Protocol (SNMP) Subagent APIs" in the book *System API Reference: UNIX-Type APIs*.
	Some other exception condition occurred.	See the messages in the job logs for both the SNMP agent job (QTCP/QTMSNMP) and the subagent job. Correct them and retry the subagent job.
The SNMP subagent works for awhile, but then pDPIpacket() returns NULL, when trying to parse a new incoming DPI packet.	There is insufficient memory to build the internal structures for the DPI packet, because it has not been freed, after having been dynamically allocated.	Ensure that the fDPIparse() routine is called so that memory use does not monotonically increase while the subagent is running. (see "Simple Network Management Protocol (SNMP) Subagent APIs" in the book *System API Reference: UNIX-Type APIs*.
	Some other exception condition occurred.	See the messages in the job logs for both the SNMP agent job (QTCP/QTMSNMP) and the subagent job. Correct them and retry the subagent job.
The SNMP subagent occasionally gets a DPI unregister or close packet from the agent with reason_code of SNMP_UNREGISTER_time-out or SNMP_CLOSE_time-out.	The subagent has taken too long to respond to some SNMP agent request. That is, longer than the time-out value used by the subagent in the mkDPIopen() or mkDPIregister() calls.	Re-open or re-register with a longer time-out, or use a smaller max_varBinds value in the mkDPIopen() call, or both. Important note: the entire SNMP agent waits for up to the time-out value for a response to each DPI request. If requests for a particular OID or subagent subtree takes a long time (relatively) for the subagent to process, then consideration should be given to registering that OID or subtree separately (by the same subagent), with a appropriately longer time-out.
The SNMP subagent gets a snmpsa_RC_err return code.	A run-time exception occurred, which is not covered by some other, more specific, return code.	See the messages in the job logs for both the SNMP agent job (QTCP/QTMSNMP) and the subagent job (that received the snmpsa_RC_err return code) for exceptions. Correct them and retry the subagent API calls. (By its nature this return code is fairly rare, and the cause for the exception is usually obvious, in either the agent or the subagent job.)

Appendix D. CL Commands for AS/400 SNMP

This section provides syntax diagrams and explanations of the CL commands for AS/400 SNMP.

ADDCOMSNMP (Add Community for SNMP) Command

Job: B,I Pgm: B,I REXX: B,I Exec



Notes:

P All parameters preceding this point can be specified in positional form.

1 A maximum of 300 repetitions.

Purpose

The Add Community for SNMP (ADDCOMSNMP) command defines an SNMP community profile and adds it to the SNMP agent community list. An SNMP agent uses a community profile to determine whether or not to honor a request sent by an SNMP manager. The community profile consists of a community name, an object access specification, and a list of the SNMP managers that are part of the community. The combination of the community name (COM) and the translate to ASCII community (ASCIICOM) parameters defines a community.

Multiple community profiles, each having a unique community name may exist in the SNMP agent community list at one time. Similarly, the same internet address may appear in more than one community profile.

The OS/400* SNMP agent does not support community views. A view is a subset of the objects in the management information base (MIB). Each OS/400 community consists of all of the objects in the MIB.

Restrictions: An SNMP manager sends three types of requests: get, get-next, and set. Get and get-next requests are used to read management information base (MIB) variables, and a set request is used to modify MIB variables. For a request from an SNMP manager to be accepted by the AS/400 SNMP agent, all of the following must be true:

1. The community name in the SNMP manager request specifies a defined community.
2. The internet address of the manager that sent the request must be listed in the community profile.
3. For a set request, the community object access must allow write operations to occur. For a get request or get-next request, read operations must be allowed.
4. For a set request, the object specified in the request must be able to be changed. For a get request or get-next request, the object must be readable.

Required Parameter

COM

Specifies the name of the SNMP community being added. Each SNMP community name must be unique.

community-name: Specify the name of the SNMP community being added. The name may contain characters that cannot be displayed (for example, X'60619E').

Optional Parameters

ASCIICOM

Specifies whether the community name is translated to ASCII characters when the community profile is added to the SNMP agent community list.

***YES:** The community name is translated to ASCII characters when the community profile is added to the SNMP agent community list. This value should be specified if the SNMP manager system defines its community names entirely of ASCII characters. An error message is sent if the community name cannot be translated to ASCII characters.

***NO:** The community name is not translated to ASCII characters when the community profile is added to the SNMP agent community list. This value should be specified if the SNMP manager system defines its community names using EBCDIC characters or characters that cannot be displayed.

INTNETADR

Specifies the internet addresses of the SNMP managers that are part of this community.

***ANY:** Allow any SNMP manager to be part of this community.

manager-internet-address: Specify the internet address of the SNMP manager. The internet address is specified in the form *nnn.nnn.nnn.nnn*, where *nnn* is a decimal number ranging from 0 through 255. An internet address is not valid if it has a value of all binary ones or all binary zeros for the network identifier (ID) portion or the host ID portion of the address. If the internet address is entered from a command line, the address must be enclosed in apostrophes. Up to 300 unique internet addresses may be specified. The same internet address may appear in more than one community profile.

OBJACC

Specifies the object access for the community.

***SNMPATR:** The object access defined with the Change SNMP Attributes (CHGSNMPA) command is used for this community.

***READ:** Allow SNMP managers that are part of this community to read all management information base (MIB) objects with get or get-next requests. Modification of MIB objects by SNMP managers is not permitted.

***WRITE:** Allow SNMP managers that are part of this community to change all MIB objects that are able to change with set requests. Specifying *WRITE implies *READ access.

***NONE:** Do not allow SNMP managers that are part of this community any access to MIB objects.

LOGSET

Specifies whether set requests from SNMP managers in this community are logged in journal QSNMP in library QUSRSYS.

***SNMPATR:** The value defined with the Change SNMP Attributes (CHGSNMPA) command is used for this community.

***YES:** Set requests are logged.

***NO:** Set requests are not logged.

LOGGET

Specifies whether get requests and get-next requests from SNMP managers in this community are logged in journal QSNMP in library QUSRSYS.

***SNMPATR:** The value defined with the Change SNMP Attributes (CHGSNMPA) command is used for this community.

***YES:** Get requests and get-next requests are logged.

***NO:** Get requests and get-next requests are not logged.

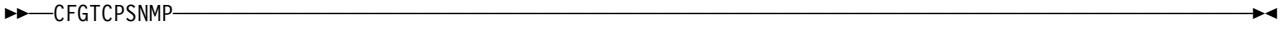
Example

```
ADDCOMSNMP   COM(ROCHESTER)
  INTNETADR('8.6.5.4' '8.6.5.3')
  OBJACC(*WRITE)
```

This command adds the community ROCHESTER to the SNMP agent community list. SNMP managers with internet addresses 8.6.5.4 and 8.6.5.3 are the only managers in the community and are able to change all MIB objects.

CFGTCPSNMP (Configure TCP/IP SNMP) Command

Job: I Pgm: I REXX: I Exec



Purpose

The Configure TCP/IP SNMP (CFGTCPSNMP) command is used to display a menu that allows a user to define or change the Simple Network Management Protocol (SNMP) configuration. The menu options include:

- Change SNMP attributes
- Work with communities for SNMP

It is not necessary to run the CFGTCPSNMP command before using the SNMP agent. The SNMP agent is shipped with a community that has the following characteristics:

Community Name public

ASCIICOM *YES

INTNETADR *ANY

OBJACC *READ

LOGSET *NO

LOGGET *NO

See the Change SNMP Attributes (CHGSNMPA) command for the default values for SNMP attributes.

There are no parameters for this command.

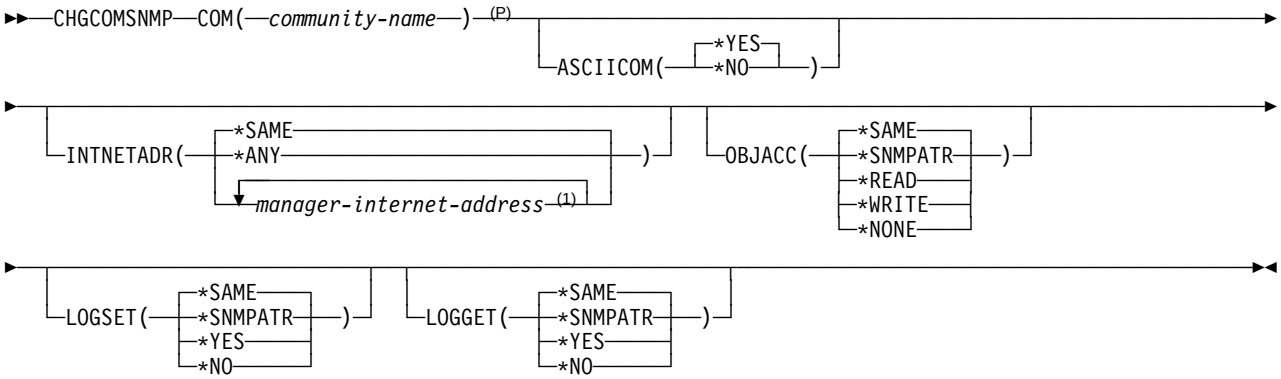
Example

```
CFGTCPSNMP
```

This command displays the Configure TCP/IP SNMP menu.

CHGCOMSNMP (Change Community for SNMP) Command

Job: B,I Pgm: B,I REXX: B,I Exec



Notes:

P All parameters preceding this point can be specified in positional form.

1 A maximum of 300 repetitions.

Purpose

The Change Community for SNMP (CHGCOMSNMP) command changes an SNMP community profile in the SNMP agent community list. An SNMP agent uses a community profile to determine whether or not to honor a request sent by an SNMP manager. The community profile consists of a community name, an object access specification, and a list of the SNMP managers that are part of the community. The combination of the community name (COM) and the translate to ASCII community (ASCIICOM) parameters defines a community.

Required Parameter

COM

Specifies the name of the SNMP community being changed. The community must already exist in the SNMP agent community list. You can define an SNMP community using the Add Community for SNMP (ADDCOMSNMP) command.

community-name: Specify the name of the SNMP community being changed. The name may contain characters that cannot be displayed.

Optional Parameters

ASCIICOM

Specifies whether the community name is translated to ASCII characters before it is compared with the community name specified in a request from an SNMP manager. This parameter is used in combination with the community name to determine the community to be changed. If this parameter is not specified and two communities have the same name but different ASCIICOM parameter values, the community that is changed is the community with ASCIICOM set to *YES.

***YES:** The community name is translated to ASCII characters before it is compared with a community name specified by an SNMP manager.

***NO:** The community name is not translated to ASCII characters before it is compared with a community name specified by an SNMP manager.

INTNETADR

The internet addresses of the SNMP managers that are part of this community.

***SAME:** The value does not change.

***ANY:** Allow any SNMP manager to be part of this community.

manager-internet-address: Specify the internet address of the SNMP manager. The internet address is specified in the form

nnn.nnn.nnn.nnn, where *nnn* is a decimal number ranging from 0 through 255. An internet address is not valid if it has a value of all binary ones or all binary zeros for the network identifier (ID) portion or the host ID portion of the address. If the internet address is entered from a command line, the address must be enclosed in apostrophes. Up to 300 unique internet addresses may be specified. The same internet address may appear in more than one community profile.

OBJACC
Specifies the object access for the community.

***SAME:** The value does not change.

***SNMPATR:** The object access defined with the Change SNMP Attributes (CHGSNMPA) command is used for this community.

***READ:** Allow SNMP managers that are part of this community to read all management information base (MIB) objects. Modification of MIB objects by SNMP managers is not permitted.

***WRITE:** Allow SNMP managers that are part of this community to change all MIB objects that can be changed. Specifying *WRITE implies *READ access.

***NONE:** Do not allow SNMP managers that are part of this community to access any MIB objects.

LOGSET
Specifies whether Set requests from SNMP managers in this community are logged in journal QSNMP in library QUSRSYS.

***SAME:** The value does not change.

***SNMPATR:** The value defined with the Change SNMP Attributes (CHGSNMPA) command is used for this community.

***YES:** Set requests are logged.

***NO:** Set requests are not logged.

LOGGET
Specifies whether get requests and get-next requests from SNMP managers in this community are logged in journal QSNMP in library QUSRSYS.

***SAME:** The value does not change.

***SNMPATR:** The value defined with the Change SNMP Attributes (CHGSNMPA) command is used for this community.

***YES:** Get requests and get-next requests are logged.

***NO:** Get requests and get-next requests are not logged.

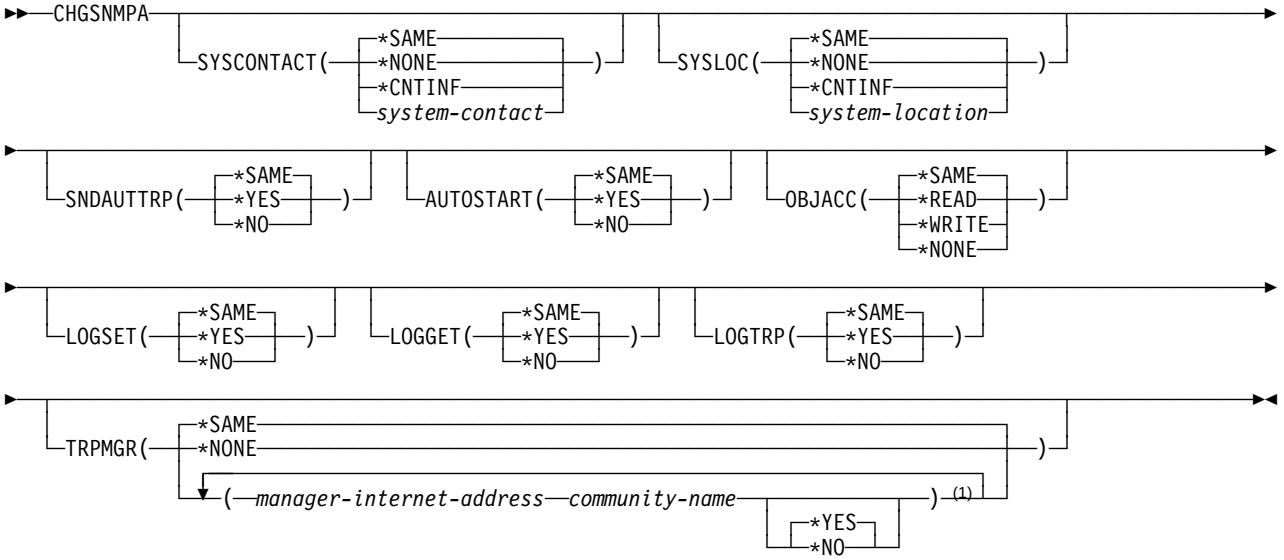
Example

```
CHGCOMSNMP   COM(ENDICOTT)   INTNETADR(*ANY)
  OBJACC(*READ)
```

This command changes community ENDICOTT to have an object access of read and to allow any SNMP manager to read the MIB objects on this system. All of the other community values are unchanged.

CHGSNMPA (Change SNMP Attributes) Command

Job: B,I Pgm: B,I REXX: B,I Exec



Note:
1 A maximum of 300 repetitions.

Purpose

The Change SNMP Attributes (CHGSNMPA) command changes values and options used by the OS/400 SNMP agent. The command also is used to specify which SNMP managers receive traps generated by the local AS/400 system.

The SNMP agent is shipped with the following values for the SNMP attributes.

Keyword	Value
SYSCONTACT	*NONE
SYSLOC	*NONE
SNDAUTTRP	*YES
AUTOSTART	*NO
OBJACC	*READ
LOGSET	*NO
LOGGET	*NO
LOGTRP	*NO
TRPMGR	*NONE

Optional Parameters

SYSCONTACT
Specifies the name of the contact person for this AS/400 system, along with information on how to contact this person. This value is used only by SNMP-specific functions. This value also may be read or modified by an authorized SNMP manager.

***SAME:** The value does not change.

***NONE:** No system contact exists.

***CNTINF:** The value is obtained from the service contact information specified by using the Work with Contact Information (WRKCNTINF) command. The value obtained consists of the contact person and the contact telephone numbers.

system-contact: Specify the name of the contact person and other contact information. All of the characters specified must be able to be translated into the ASCII character set.

SYSLOC
Specifies the physical location of this AS/400 system. This value is used only by SNMP-specific functions. This value also may be read or modified by an authorized SNMP manager.

***SAME:** The value does not change.

***NONE:** No system location information exists.

***CNTINF:** The value is obtained from the service contact information specified by using the Work with Contact Information (WRKCNTINF) command. The value obtained consists of the mailing address.

system-location: Specify the physical location of the system. All of the characters specified must be able to be translated into the ASCII character set.

SNDAUTTRP

Specifies whether the SNMP agent may send any authenticationFailure traps to any defined SNMP managers. An authenticationFailure trap is sent by the SNMP agent if a request is received from an SNMP manager that contains a community name that is not recognized by the SNMP agent. This trap is only sent when SNDAUTTRP is *YES and when at least one trap manager has been defined. This value may also be read or modified by an authorized SNMP manager.

***SAME:** The value does not change.

***YES:** authenticationFailure traps may be sent.

***NO:** authenticationFailure traps are not sent.

AUTOSTART

Specifies whether the SNMP agent is started when the STRTCP command runs.

***SAME:** The value does not change.

***YES:** The SNMP agent is started when the STRTCP command runs.

***NO:** The SNMP agent is not started when the STRTCP command runs.

OBJACC

Specifies the default object access for SNMP communities.

***SAME:** The value does not change.

***READ:** Allow SNMP managers that are part of a community to read all management information base (MIB) objects. Modification of MIB objects by SNMP managers is not permitted.

***WRITE:** Allow SNMP managers that are part of a community to modify all MIB objects that can be modified. Specifying *WRITE implies *READ access.

***NONE:** Do not allow SNMP managers that are part of a community to modify any MIB objects.

LOGSET

Specifies the default value for whether set requests from SNMP managers in a community are logged in journal QSNMP in library QUSRSYS.

***SAME:** The value does not change.

***YES:** Set requests are logged.

***NO:** Set requests are not logged.

LOGGET

Specifies the default value for whether get requests and get-next requests from SNMP managers in a community are logged in journal QSNMP in library QUSRSYS.

***SAME:** The value does not change.

***YES:** Get requests and get-next requests are logged.

***NO:** Get requests and get-next requests are not logged.

LOGTRP

Specifies whether traps are logged in journal QSNMP in library QUSRSYS.

***SAME:** The value does not change.

***YES:** Traps are logged.

***NO:** Traps are not logged.

TRPMGR

Specifies which SNMP managers receive traps generated by this AS/400 system.

***SAME:** The value does not change.

***NONE:** No SNMP managers receive traps.

Element 1: Manager Internet Address

manager-internet-address: Specify the internet address of the SNMP manager. The address must be of the form *nnn.nnn.nnn.nnn*, where *nnn* is a decimal number ranging from 0 to 255. This address is independent of the manager internet address specified on the ADDCOMSNMP and CHGCOMSNMP commands.

Element 2: Community Name

community-name: Specify the SNMP community name to be placed in the traps sent to this SNMP manager. The community name specified in this element is independent of the community name specified on the ADDCOMSNMP, CHGCOMSNMP, and RMVCOMSNMP commands. The name may contain characters that cannot be displayed.

Element 3: Translate Community Name

***YES:** The community name is translated to ASCII characters when a trap is sent to the SNMP manager. This value should be specified when the community name consists entirely of characters that can be displayed. An error message is sent if the community name cannot be translated to ASCII characters.

***NO:** The community name is not translated to ASCII characters when a trap is sent to the SNMP manager. This value should be specified when the community name contains one or more characters that cannot be displayed.

Examples

Example 1: Changing System Contact and Automatic Start

```
CHGSNMPA   SYSCONTACT('JOE SMITH, PHONE 555-1212')
  AUTOSTART(*NO)
```

This command changes the system contact information and specifies that the SNMP agent should not start when the STRTCP command runs. All other values are unchanged.

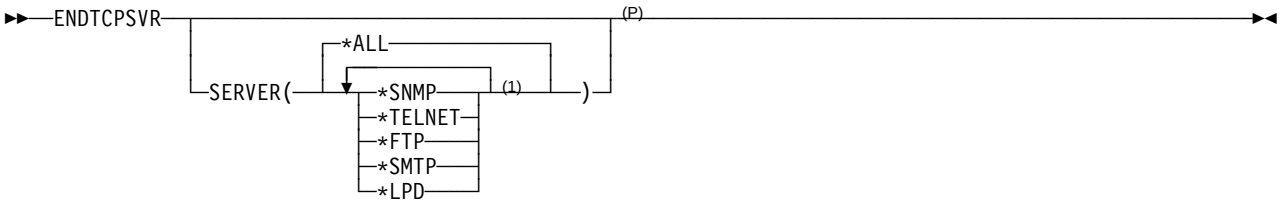
Example 2: Changing Trap Managers

```
CHGSNMPA   TRPMGR(('9.8.7.6' 'TRAPCOMMUNITY')
  ('9.8.7.5' 'TRAPCOMMUNITY2'))
```

This command causes any traps generated by the local AS/400 system to be sent to SNMP managers that have internet protocol addresses 9.8.7.6 and 9.8.7.5. Community name TRAPCOMMUNITY is placed in traps sent to 9.8.7.6, and community name TRAPCOMMUNITY2 is placed in traps sent to 9.8.7.5. For both managers the community name is translated to ASCII characters before being placed in the trap.

ENDTCPSVR (End TCP/IP Server) Command

Job: B,I Pgm: B,I REXX: B,I Exec



Notes:
1 A maximum of 5 repetitions.

P All parameters preceding this point can be specified in positional form.

Purpose

The ENDTCPSVR command is used to end the TCP/IP application server jobs that are specified in the SERVER parameter. If the jobs have any current active connections, these connections are ended immediately. If the ENDTCPSVR command is used to end a server that is not active, a diagnostic message is returned.

Optional Parameter

SERVER

Specifies which of the TCP/IP application server jobs is to be ended by this command.

***ALL:** All of the TCP/IP server jobs are ended.

***SNMP:** All jobs associated with the SNMP agent in the QSYSWRK subsystem are ended.

***TELNET:** All TELNET server jobs are ended.

***FTP:** All FTP server jobs are ended.

***SMTP:** All jobs associated with SMTP in the QSYSWRK subsystem are ended. The bridge job in the QSNADS subsystem is not ended.

***LPD:** All LPD server jobs are ended.

Examples

Example 1: Ending All TCP/IP Servers

```
ENDTCPSVR
```

This command ends all active TCP/IP application server jobs running in the QSYSWRK subsystem.

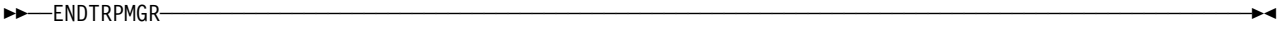
Example 2: Ending the LPD Servers

```
ENDTCPSVR   SERVER(*LPD)
```

This command ends the TCP/IP LPD application server jobs.

ENDTRPMGR (End Trap Manager) Command

Job: B,I Pgm: B,I REXX: B,I Exec



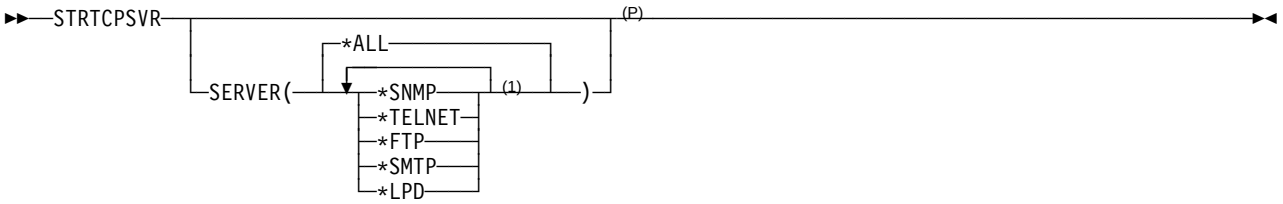
Purpose

The End Trap Manager (ENDTRPMGR) command allows you to end the OS/400 SNMP Manager Framework trap manager job.

There are no parameters for this command.

STRTCPSVR (Start TCP/IP Server) Command

Job: B,I Pgm: B,I REXX: B,I Exec



Notes:
1 A maximum of 5 repetitions.

P All parameters preceding this point can be specified in positional form.

Purpose

The Start TCP/IP Server (STRTCPSVR) command is used to start the TCP/IP application servers that are shipped with the Operating System/400 product or the TCP/IP Connectivity Utilities/400 product. This command starts TCP/IP jobs in the QSYSWRK subsystem for the application servers specified with the server (SERVER) parameter. The number of server jobs started by this command is specified, where appropriate, in the configuration for each TCP/IP application.

All servers have an autostart (AUTOSTART) parameter on the associated configuration command (for example, Change FTP Attributes (CHGFTPA)). This parameter indicates if the server should be started when the Start TCP/IP (STRTCP) command is entered. The STRTCPSVR command ignores the value of a server's autostart parameter.

Optional Parameter

SERVER

Specifies the TCP/IP application servers to be started by this command.

***ALL:** All of the TCP/IP application servers are started.

***SNMP:** The Simple Network Management Protocol (SNMP) agent jobs are started. Subsequent usage of the STRTCPSVR SERVER(*SNMP) command results in a diagnostic message if the SNMP server jobs have already been started.

***TELNET:** The TELNET server is started. Subsequent usage of the STRTCPSVR SERVER(*TELNET) command starts one additional TELNET server.

Note: Having more than one TELNET server job running reduces the chances of having connection attempts refused.

***FTP:** The File Transfer Protocol (FTP) servers are started based on the number of servers configured with the Change FTP Attributes (CHGFTPA) command. Subsequent usage of the STRTCPSVR SERVER(*FTP) command starts one additional FTP server.

Note: Having more than one FTP server job running can improve the performance of initiating a session when multiple users attempt to connect to the server in a short period of time.

***SMTP:** The Simple Mail Transfer Protocol (SMTP) client and server jobs are started. Additional SMTP client and server jobs cannot be started. Subsequent usage of the STRTCPSVR SERVER(*SMTP) command results in a diagnostic message if the SMTP server jobs have already been started.

***LPD:** The line printer daemon (LPD) servers are started based on the number of servers configured with the Change LPD Attributes (CHGLPDA) command. Subsequent usage of the STRTCPSVR SERVER(*LPD) command starts one additional LPD server.

Note: LPD works most efficiently when two or more servers are running. Running only one server works, but no jobs can

be received while a current job is running. If a large print job is running, new jobs have to wait before LPD is ready to accept any new line printer requester (LPR) requests.

Examples

Example 1: Starting all TCP/IP Servers

```
STRTCPSVR
```

This command starts all of the TCP/IP application servers that have been configured to be started. For example: If the Change FTP Attributes (CHGFTPA) command had previously been used to configure two FTP servers, both servers would be started when STRTCPSVR is issued. This example is also true for other TCP/IP application servers.

Where appropriate, the number of servers to start is based on the number of servers configured for the server being started. The configuration option to automatically start the servers (AUTOSTART) is ignored by the STRTCPSVR command. The AUTOSTART parameter is used only by the STRTCP command.

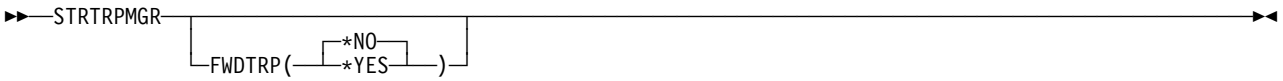
Example 2: Starting the TELNET Server

```
STRTCPSVR  SERVER(*TELNET)
```

This command starts the TCP/IP TELNET application server. If the TELNET server had been previously started, one additional TELNET server job would be started.

STRTRPMGR (Start Trap Manager) Command

Job: B,I Pgm: B,I REXX: B,I Exec



Purpose

The Start Trap Manager (STRTRPMGR) command allows you to start the OS/400 SNMP Manager Framework trap manager job. An optional Forward Trap parameter may be specified which enables traps that are received on other systems to be forwarded to other Network Management stations. The trap manager uses the trap generation and sending facilities provided in the Simple Network Management Protocol (SNMP) agent and Distributed Protocol Interface (DPI) interface.

Optional Parameters

FWDTRP

Specifies whether traps received on the system are to be forwarded to other network management stations.

***YES:** Received traps are forwarded using the facilities provided in the SNMP agent and DPI interface.

***NO:** Received traps are not forwarded. Traps are only enqueued.

Examples

Example 1: Start Trap Manager Job (Enqueue Traps Only)

```
STRTRPMGR
```

This command starts the trap manager job. Traps received by the trap manager are enqueued only.

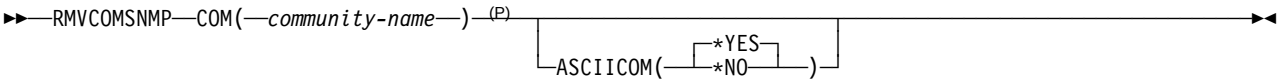
Example 2: Start Trap Manager Job (Enqueue & Forward Traps)

```
STRTRPMGR FWDTRP(*YES)
```

This command starts the trap manager job. Traps received by the trap manager are enqueued and forwarded.

RMVCOMSNMP (Remove Community for SNMP) Command

Job: B,I Pgm: B,I REXX: B,I Exec



Note:
P All parameters preceding this point can be specified in positional form.

Purpose

The Remove Community for SNMP (RMVCOMSNMP) command is used to remove a Simple Network Management Protocol (SNMP) community profile from the SNMP agent community list. The community profile consists of a community name, an object access specification, and a list of the SNMP managers that are part of the community. The combination of the community name (COM) and the translate to ASCII community (ASCIICOM) parameters defines a community.

Required Parameter

COM

Specifies the name of the SNMP community being removed. The community must already exist in the SNMP agent community list.

community-name: Specify the name of the SNMP community being removed. The name may contain characters that cannot be displayed.

Optional Parameter

ASCIICOM

Specifies whether the community name is translated to ASCII characters before it is compared with the community name specified in a request from an SNMP manager. This parameter is used in combination with the community name to determine the community to be removed. If two communities have the same name and you don't specify the ASCIICOM parameter, the community with ASCIICOM set to *YES is removed.

***YES:** The community name is translated to ASCII characters before it is compared with a community name specified by an SNMP manager.

***NO:** The community name is not translated to ASCII characters before it is compared with a community name specified by an SNMP manager.

Example

```
RMVCOMSNMP   COM(ROCHESTER)
```

This command removes community ROCHESTER from the SNMP agent community list.

Bibliography

If you want more information on a topic while you are using this book, see the *Publications Reference* book, SC41-5003, and the *Advanced 36 Information Directory* book, SC21-8292, for related AS/400 publications.

The following publications provide additional information about the topics described or referred to in this book. The books are listed with their full titles and order numbers. When AS/400 books are referred to in this book, a shortened version of the title is used.

IBM Publications

Communications and Programming

The following IBM AS/400 publications provide additional information about topics described or referred to in this book:

- *APPN Support*, SC41-5407, provides information about the concepts of AS/400 advanced peer-to-peer networking (APPN) and about planning APPN networks.
- *Backup and Recovery* book, SC41-5304, provides information to help you become familiar with AS/400 functions, develop a backup plan, and recover from system failures.
- *Communications Management*, SC41-5406, contains information about operating communications and handling communications errors.
- *Communications Configuration*, SC41-5401, contains general configuration information, including descriptions of network interface, line, controller, device, modes and class-of-service descriptions. Information about configuration lists and connection lists is also included.
- *CL Programming*, SC41-5721, provides a discussion of AS/400 programming topics, such as a general discussion of objects and libraries, control language (CL) programming, messages and message handling, user-defined commands and menus, and application testing.
- *CL Reference*, SC41-5722, provides a description of the AS/400 control language (CL) and its commands.
- *ILE C/400 Programmer's Guide*, SC09-2069 and *ILE C/400 Programmer's Reference* book, SC09-2070, Provides information on how to develop applications using the ILE C/400 language. Includes information about creating, running and debugging programs. Also includes programming considerations for interlanguage program and procedure calls, locales, handling exceptions, database, externally described and device files.
- *System API Reference: UNIX-Type APIs* book, SC41-5875, and *System API Reference: Client Support APIs* book, SC41-5851, provide descriptions of the OS/400 application programming interfaces (APIs).
- *TCP/IP Configuration and Reference* book, SC41-5420, provides information for configuring and using AS/400 TCP/IP support. The applications included are Network Status (NETSTAT), Packet InterNet Groper (PING), TELNET, File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP), line printer requester (LPR), and line printer daemon (LPD). The TCP and UDP Pascal application program interface (API) is also discussed.

NetView

- *Learning About NetView: Network Concepts*, SK2T-0292 (PC Diskette)
- *NetView Administration Reference*, SC30-3361
- *NetView Command Lists*, SC30-3423
- *NetView Command Summary*, SX27-3620
- *NetView Customization*, LY30-5586
- *NetView Diagnosis*, LY30-5587
- *NetView Hardware Problem Determination Reference*, SC30-3366
- *NetView Installation and Administration* book, SC30-3360
- *NetView Messages*, SC30-3365
- *NetView Operation*, SC30-3364
- *NetView Operation Primer*, SC30-3363
- *NetView Operation Scenarios*, SC30-3376
- *Network Program Products Bibliography and Master Index*, GC30-3353
- *Network Program Products General Information*, GC30-3350
- *Network Program Products Planning*, SC30-3351
- *Network Program Products Samples: NetView*, SC30-3352
- *Network Program Products Storage Estimates*, SC30-3403

Advanced Communications Function for Virtual Telecommunications Access Method (ACF/VTAM)

- *ACF/VTAM* General Information*, GC38-0254
- *ACF/VTAM System Programmer's*, SC38-0258

Systems Network Architecture (SNA)

- *Systems Network Architecture Technical Overview*, GC30-3072
- *Systems Network Architecture Formats*, GA27-3136
- *Systems Network Architecture Format and Protocol Reference*, SC30-3112
- *Systems Network Architecture–Sessions Between Logical Units*, GC20-1868

Non-IBM Publications

Simple Network Management Protocol

- Case, J. D., Fedor, M. S., Schoffstall, M. L., and Davin, J. R. *A Simple Network Management Protocol. RFC 1157*, SNMP Research, May 1990.
- Case, J. D., Davin, J. R., Fedor, J. R., and Schoffstall, M. L. *Network Management and the Design of SNMP*, *ConneXions--The Interoperability Report*, 3(3):22-26, March 1989, ISSN 0894-5926.
- Rose, M. T., and McCloghrie, K. *Concise MIB Definitions. RFC 1212*, Performance Systems International, Inc., March 1991.
- Rose, M. T. *Network Management is Simple: You just need the "Right" Framework.* In *Integrated Network Management, II*, pages 9-25, IFIP WG 6.6, North Holland, April 1991. ISBN 0-444-89028-9.
- Rose, M. T. *The Simple Book: An Introduction to Internet Management*, Englewood Cliffs:P T R Prentice-Hall, 1994.

Index

E

F

G

I

L

M

N

O

P

Q

R

S

T

U

V

W

Reader Comments—We'd Like to Hear from You!

AS/400
Simple Network Management
Protocol (SNMP) Support
Version 4

Publication No. SC41-5412-00

Overall, how would you rate this manual?

	Very Satisfied	Satisfied	Dissatis-fied	Very Dissatis-fied
Overall satisfaction				

How satisfied are you that the information in this manual is:

Accurate				
Complete				
Easy to find				
Easy to understand				
Well organized				
Applicable to your tasks				
THANK YOU!				

Please tell us how we can improve this manual:

May we contact you to discuss your responses? __ Yes __ No
Phone: (____) _____ Fax: (____) _____ Internet: _____

To return this form:

- Mail it
- Fax it
  United States and Canada: **800+937-3430**
  Other countries: **(+1)+507+253-5192**
- Hand it to your IBM representative.

Note that IBM may use or distribute the responses to this form without obligation.

Name

Address

Company or Organization

Phone No.

Reader Comments—We'd Like to Hear from You!
SC41-5412-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 542 IDCLERK
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-9986

Fold and Tape **Please do not staple** Fold and Tape

SC41-5412-00

Cut or Fold
Along Line

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC41-5412-00

Spine information:

AS/400

Simple Network Management Protocol (SNMP) Support

Version 4