IBM

3Com

IBM/3Com Heterogeneous LAN Management Architecture Reference Version 0.1.0

SC30-3539-0

IBM/3Com Heterogeneous LAN Management Architecture Reference Version 0.1.0

Document Number SC30-3539-0

December 3, 1990

PREPUBLICATION COPY

This preliminary publication is based on present knowledge of the subject, and specifications contained herein are subject to change. All preliminary publications become obsolete upon availability of the final publication.

Preliminary publications are distributed only when necessary and only to immediately concerned individuals. Recipients incur responsibility for usage of this material and are required to exercise good judgment when permitting others to read it.

Note: Sections of this document have been submitted to Project 802 of the IEEE. These sections may change to reflect the work being done by that standards body.

3Com

3Com Corporation
5400 Bayfront Plaza
P.O. Box 58145
Santa Clara, CA 95052-8145

(408) 764-5000
345546 Telex

December 26, 1990

Thank you for your interest in the 3Com/IBM jointly developed specifications for Heterogeneous LAN Management (HLM). Please find enclosed draft copies of the Architectural Reference and the API Technical Reference documents describing HLM.

Included with the specifications, please also find a brief monograph entitled '*Heterogeneous LAN Management, a Joint 3Com/IBM Architectural Proposal*'. This monograph supplements the HLM Specification documents, and it will provide you with a general overview of the HLM approach.

Questions and comments regarding these specifications are welcomed, but given the scope of the draft, we would appreciate all inquiries be submitted in writing. Please send your comments to:

3Com Corporation
5400 Bayfront Plaza
Santa Clara, Calif. 95052
Atten: Jim Healey, NAD Marketing

To assist us in addressing your questions, please be sure to include a telephone number with your written comments, so that we can contact you if necessary.

Once again, thank you for your interest in the HLM specifications.

Jim

Jim Healey
Product Marketing Manager
Network Adapter Division, 3Com

Heterogeneous LAN Management Architecture Reference Review Form

Reviewer's Section

Reviewer's Name: **Company:**

Specifications Page Number(s):

Section Heading(s):

Comment or Concern:

Recommendation:

IBM/3Com Section

Response:

Comment Tracking Number:

Heterogeneous Lan Management(HLM), a Joint 3Com/IBM Architectural Proposal

by

Richard Watson
Enabling Software Manager, Network Adapter Division

Paul Sherer
Architect, Network Adapter Division

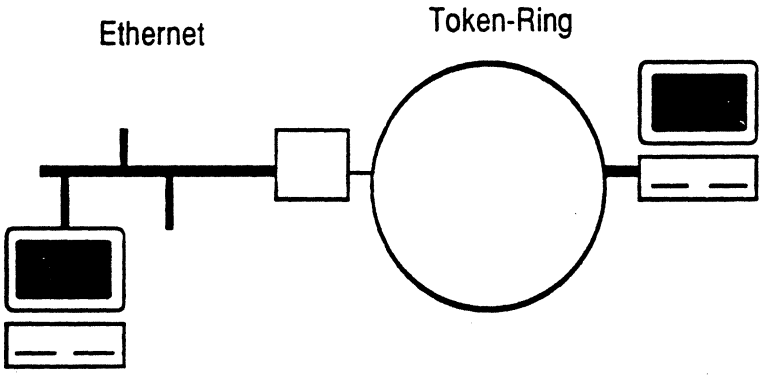
3Com Corporation
Santa Clara, California

October 1990

Abstract:

Addressing today's diverse network management requirements is made more complex considering that most major LAN installations have a mix of media types and operating system environments. This heterogeneous configuration posses special challenges to providing a uniform management architecture, but it is vital that these issues be addressed. The Heterogeneous Lan Management(HLM) architecture is a set of architectural documents that have resulted from a joint study done by 3Com and IBM to define a management platform that will satisfy the management needs of mixed Ethernet and Token-Ring environments regardless of the operating system supporting each LAN node.

This article is targeted to the ISV developers, NDIS developers, and LAN system administrators.



I. Introduction

A. BACKGROUD

Network Management is one of the hottest issues and most mis-understood concepts in the local area network business. A great deal of confusion exists as to exactly what network management services are required and who will supply those services. Most major 'players' (e.g., IBM, DEC, 3Com, Hewlett-Packard, ATT&T, etc..) are creating offerings in their respective domains of business, but, these solutions tend to be vendor specific and more specifically, LAN media

specific. Solutions for one specific vendor's product offering does not meet today's diverse LAN environment management needs.

Heterogeneous lan installations are not a future consideration, but exist in today's Fortun 1000 corporation. According to a recent International Data Corporation (IDC) bulletin, based on a survey conducted by *Network World* and Installed Technology International (ITI), the number of concurrent installations of both Ethernet and Token-Ring technologies is highly related to the type of vertical industry and company size.

Of the 24,000 contacts at 12,046 sites responding to the survey, 19 percent reported having both Ethernet and Token Ring installed. This compares with 50 percent of the total sites with Ethernet only and 36 percent of total sites with Token Ring only. As the data below shows, the Transportation/Utilities section has by far the highest penetration, with 27 percent. This is not surprising, given this sector's high Ethernet (52 percent) and high Token Ring (45 percent) penetration. No one LAN media has a clear domination in the market. It appears that for the foreseeable future, mixed (or heterogeneous) LANs will be a fact for the larger businesses.

In addressing the broad spectrum management issue, the international standards bodies, have focused on the definition of communications protocols. These protocols provide standard ways to send management information between the nodes on a network. Familiar protocols include the Open Systems Interconnect's (OSI) Common Management Information Protocol (CMIP) and the Simple Network Management Protocol (SNMP), defined by the Internet Engineering Task Force (IETF).

While adherence to one of these protocols means a vendor uses a standard way to communicate its management information, it does not mean that multi-vendor interoperability is ensured. In addition, these standard management protocols are not appropriate for management of such memory-constrained devices as networked DOS-based personal computers, which are prevalent in today's homogeneous and mixed networks.

Networks are increasingly essential to the day-to-day activity of the corporate enterprise. A network manager is typically responsible for the workgroup (including network cabling, personal computer workstations and basic network services), internetwork devices (including bridges, brouters and gateways) and all other network-connected data processing equipment. Network management can be thought of as the tools and techniques that organizations employ to ensure the continued proper operation of a network.

The many of these networks are either pure Ethernet or Token Ring, but a growing number comprise a combination of the two technologies. The common denominator is a need to manage it all, resulting in a reduction in LAN failures and downtime.

B. THE PROBLEM

While many management products are available in today's market, none todate provide the total solution to the network administrator who is challenged with managing a network comprised of mixed physical media(Ethernet/Token-Ring) and mixed operating system (DOS, OS/2, Unix, etc...). In order to address the

management issues in such a configuration, multiple products are often necessary to administer each sub-component requiring an inordinate amount of administration time and effort without meeting the full requirements.

In order to provide the proper management solution for mixed media Lan installations, a "unified" solution must be available. Unified in the sense that the administrator is required to interface with only one management system to achieve the necessary control, regardless of the media or operating system environment. To achieve such a solution necessitates the cooperation of industry wide contributors.

C. APPROACHING A SOLUTION

3Com and IBM are the leading vendors for their respective LAN media products (Ethernet, Token-Ring), and both companies share a common customer problem of manageability across mixed media. It was this common "problem" that brought these two companies together to propose an architectural solution. On July 9, 1990 both company jointly announced their combined efforts to create the Heterogeneous Lan Management (HLM) architecture.

HLM architecture is designed to be independent of network operating systems and to provide management of devices on mixed-media LANs, particularly those in which memory is constrained. HLM consists of three basic components: 1) a network management protocol common to Ethernet, Token Ring and other network configurations; 2) application programming interfaces (APIs) through which developers can create compatible network management software; and 3) specifications of what management data is accessible.

Definition of the HLM architecture is an attempt to provide a solution to network management across mixed environments and is expected to have broad industry acceptance, because the need is today!

II. HLM Architecture

A. Requirements -

Since both 3Com and IBM are manufacturers of popular network attachment products, an important aspect of the HLM architecture is directed to management support of the "lower" level set of network services. This "bottom's up" view of the LAN readily provided a very practical set of management services that has applicability to any media type and any operating system environment. Understanding the goals of HLM is key to understanding the base architecture.

HLM Design Goals

* To manage all types end stations(DOS, OS/2, Unix, etc...) in addition to concentrators, hubs, routers, bridges, and servers.

Defining an architecture that has the flexibility to provide a set of minimum management services that would permit simple extensions for management of more complex environments was one of the initial design goals. Specifically, providing a set of meaningful management services to memory constrained DOS system was

one of the base line requirements. Any successful network management product must provide management access to DOS stations while imposing the minimum impact on the application memory environment. This means developing a clear definition on what constitues the "minimum" set of managed services. Since the physical and logical link layers are vital to any LAN attached device, the HLM management support for OSI layers 1 and 2(Physical and Data Link) are defined as common for all HLM compliant systems.

* Provide architecture to manage Ethernet and Token-Ring today and others tomorrow.

Defining a management architecture for Ethernet and Token-Ring is the current requirement, but LAN technology continues to evolve and HLM must be flexible enough to accommodate new media types (e.g., FDDI). As expressed in the HLM design, the interoperability between ethernet and Token-Ring is dependent on adherence to established industry standards. This design basis will provide a platform for support of future media types.

* Provide minimal management services that can be supported by any network node, with extensions to accommodate vendor and user extended management entities.

While DOS memory constraints may place a practical limit on the minimum management services defined, other popular operating systems (OS/2, Unix, etc...) do not share the same constraints. In order to extend HLM's effectiveness, a mechanism was necessary to permit user or vendor extensions that would interoperate with the base HLM products. In this way management of other systems or application level entities could be provided.

* Be a standards based architecture.

Perhaps most important was a goal to follow current and proposed management standards. This posture, more clearly assures acceptance and future system interoperabiliity than by creating another proprietary management architecture.

B. Implementation -

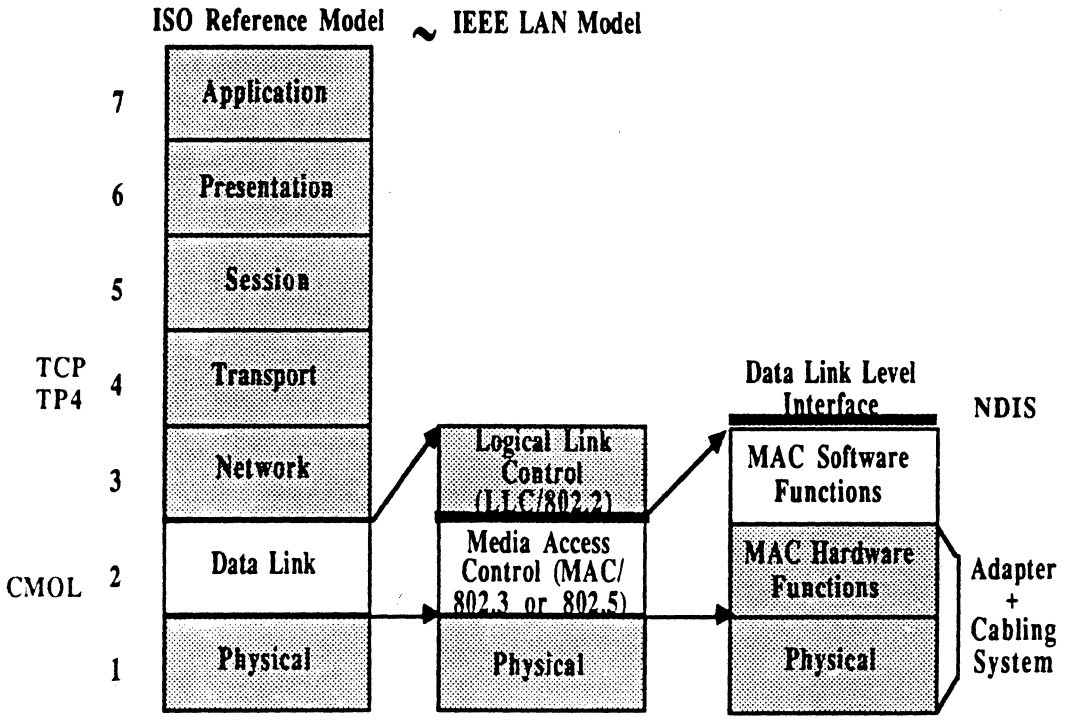
The result of the 3Com/IBM joint study agreement is not product, but rather a set of three architectural documents. The first document is the HLM Architecture Specification that provides the architectural framework for the system in describing the PDUs, protocol elements of procedure, and MIB definitions. The second document, HLM API Technical Reference, describes the necessary functional sub-components to support the HLM environment. The third document details the LLC (802.2) functions necessary to fit in the HLM framework. All these documents are available to the public from either 3Com or IBM and are not proprietary to any one company.

Base Management Protocol

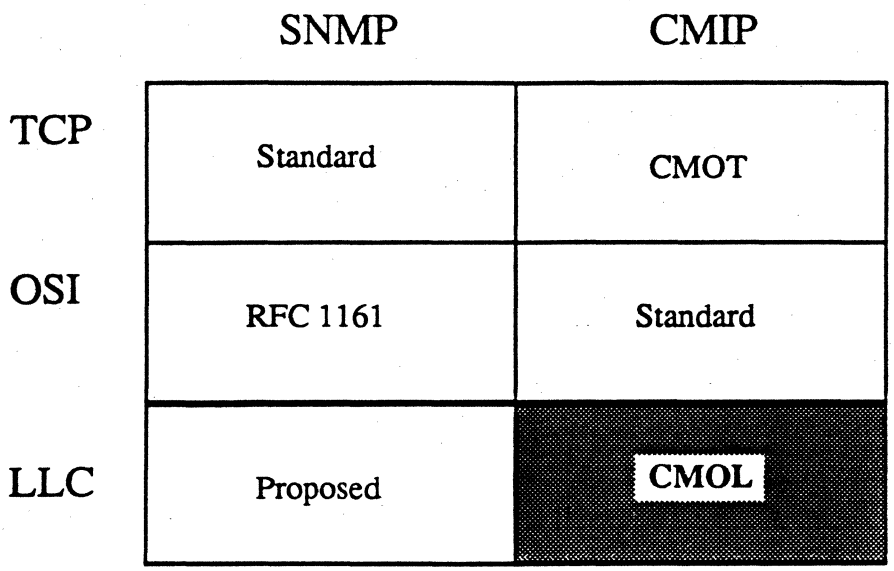
As the architecture for HLM was developed based on the initial goals, several major architectural components were developed and described in the documents. Key to

meeting one of the initial goals was the concept of providing DOS workstation management in a "standards" framework. This was accomplished by specification of the OSI CMIP standard utilizing a lower level network service(LLC). The figure below illustrates how it works, where the full TCP and OSI stacks occupy considerable memory space as they provide networking functions through the Transport Layer (TCP at layer 4) and the Application Layer (OSI at all seven layers) of the OSI model.

CMOL - CMIP over LLC

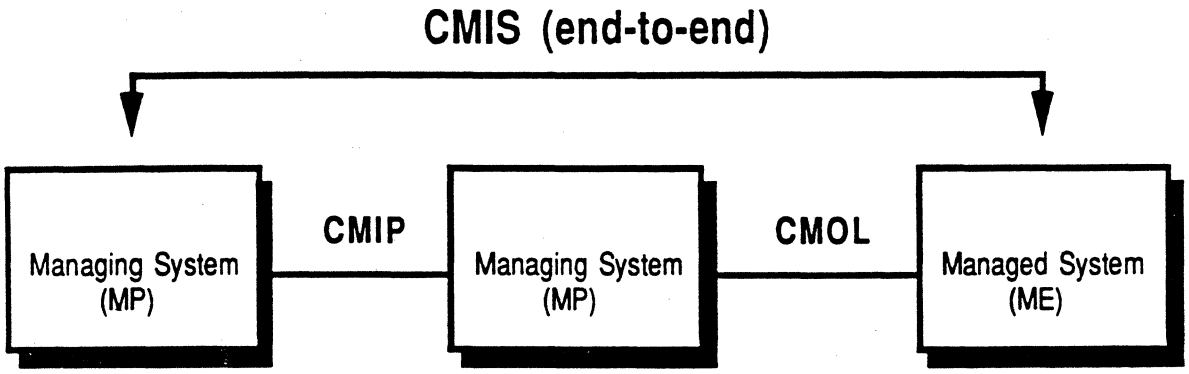


The LLC and Media Access Control (MAC) sublayers of the IEEE model correspond to the Data Link Layer (layer 2) of the OSI model. The Logical Link Control (LLC) sublayer is an IEEE standard, heavily implemented and enhanced by IBM and other vendors, which defines how command packets are generated and interpreted for support of the logical link functions of one or more logical links.



CMOL (CMIP over LLC) incorporates CMIP on top of the widely implemented LLC protocol for compatibility with existing mixed-media network installations. Since a large portion of the CMIS interface is implemented at layer 2 rather than layer 4, CMOL will not require a full OSI stack. This leaves additional memory for running other applications on such devices as DOS-based personal computers and internetworking equipment.

Specification of CMOL also fits with the basic OSI approach of having full CMIP services between hierachical management stations and allowing n-stations to be managed by some "sub-set" of CMIP(see below). In this architecture, consistent CMIS services are provided across the entire network, while meeting the specific implementation requirements imposed by memory constrained systems.



It is anticipated that HLM's network management protocol, CMOL, will be implemented in a variety of operating system environments. For DOS and OS/2 environments, portions of the initial implementation will be based on proposed extensions to the 3Com-Microsoft NDIS.

HLM Functions

Lan Station Manager(LANSM)

The base HLM services are provided by an entity described as the Lan Station Manager(LANSM). The functions provided by this entity will be common to all HLM stations, regardless if managing or being managed. It is this module that will support such base functions:

1. Discovery/Registration
2. CMOL processing
3. Base MIB services
4. Extended API support

In Discovery/Registration, each participating network station may determine the managing "environment" of the LAN and proceed to "register" itself with the appropriate manager. Machanisms have been described to accommodate the asynchronous nature of network devices and there is no order of initiation required for either managing or managed stations.

All the basic CMIP/LLC and ROSE(Remote Operations Services Entity) processing will be performed by the LANSM and is responsible for interfacing with the HLM transport system for communications with other HLM entities. No sub-elements of the HLM architecture (e.g., Layer Management Entities(LME)) will be required to process and handle the CMOL transactions.

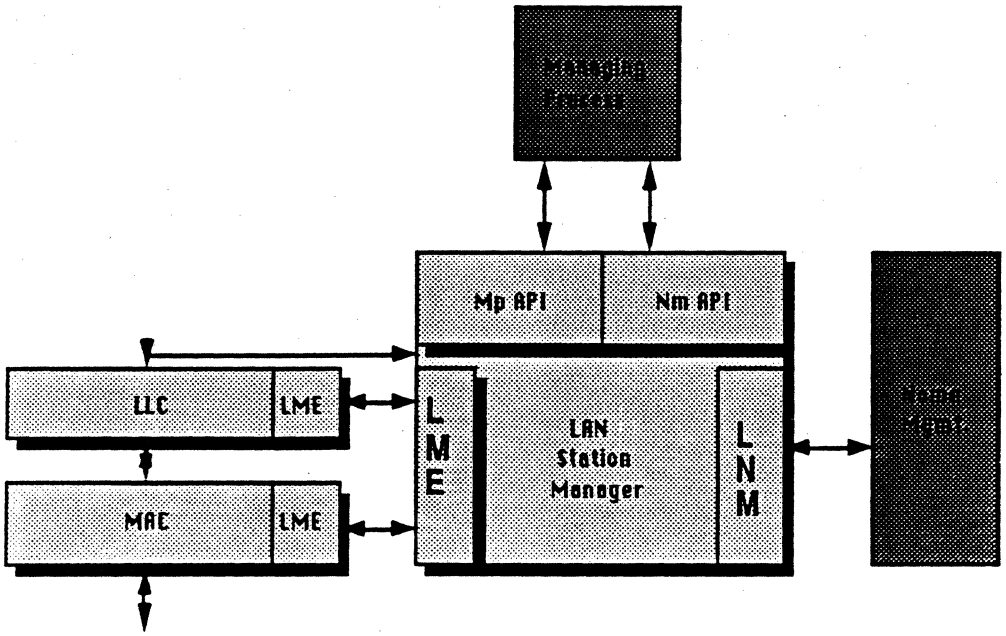
Providing the "minimum" set of management services is also handled by the LANSM by providing management access to the MAC, LLC, and "environment" MIBs. The LANSM will interface with the appropriate LME to respond to HLM management commands directed at these three functions. The MIB information for the MAC and LLC are supplied by the appropriate software component while the "environment" MIB is supported directly by the LANSM. The "environment" MIB consists of information pertaining to describing the stations and its current system configuration state(DOS I.D., amount of memory, location I.D., etc...).

Support of the other HLM functions is provided by the LANSM in exposing a set of API services. These services provide a programmatic link into the HLM management system that support generation of Managing applications/process(MP) and vendor extended Managed Enities(ME).

Managing Process(MP)

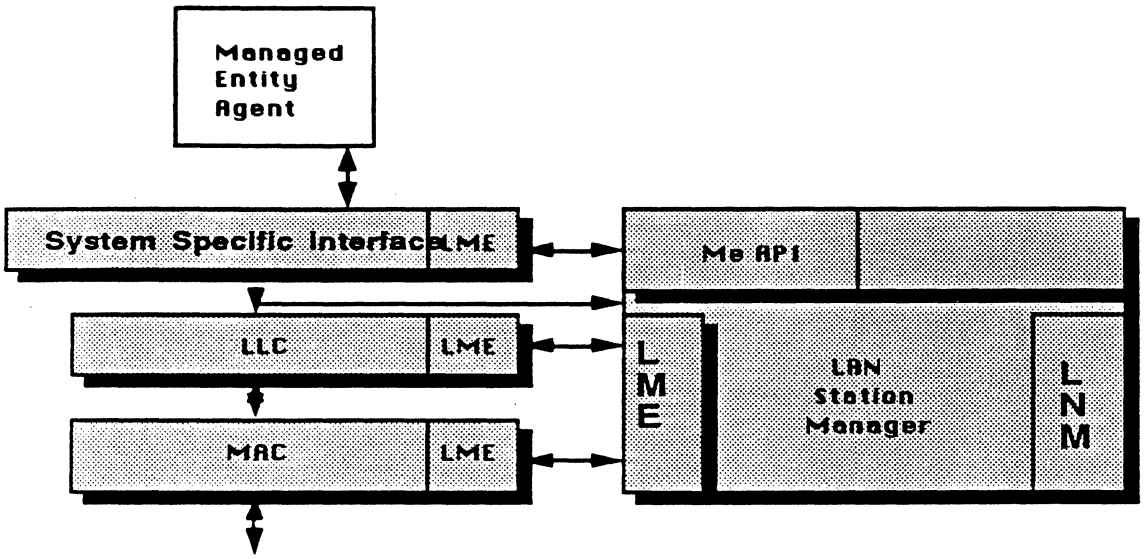
A managing process(MP) provides the user interface control for managing services as permitted in the CMIP framework. While HLM does not describe the user interface functionality, it does describe the base control functionality as supported through the API services provided by the LANSM. Using these services(described in the HLM API Technical Reference), a managing process may identify and control all HLM stations that may be present on the LAN. The scope of the management process function may be set by the management application developer, in that manager may be focused on one layer or may encompass control over all management layers supported. In additions, the architecture does allow for multiple

managers to coexist. The figure below depicts the basic relationship between the managing process(MP) and the LANSM.



Managed Entity(ME)

A Managed Entity(ME) is any managed MIB or functional domain that is registered with an HLM manager. Several "internal" or inferred MEs are architected into HLM; 1) Stations "Environment" LME; 2) LLC LME; and 3) MAC LME. These MEs provide the minimum management service base as described in the HLM documents. Extended management services may be generated via use of the Managed Entity APIs provided through the LANSM. These API services allow creation of extended managed objects(MIBs) that may be vendor or application specific. In systems that permit the added memory requirements, other system level MEs or even applications MEs may developed and execute simultaneously on a single workstations. The figure below depicts the basic relationship between the ME and the LANSM.



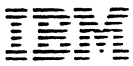
[Wrap up paragraph goes here]

III. Future of HLM

* Providing a common basis for network management that is vendor and media independent.
* How will it be developed?
* Where does NDIS fit in?
* IEEE acceptance
* Future product announcements from both 3Com and IBM
* Impact on Lan administrative functions(integration in Overlord)
* broad industry acceptance(Novell, Microsoft, SCO, Banyan, etc...)

Possible SIDE BARS --

1. CMIP vs. CMOL
2. Why not SNMP



IBM/3Com Heterogeneous LAN Management Architecture Reference Version 0.1.0

SC30-3539-0

Special Notices

The following items used in this publication, denoted by an asterisk (*), are registered trademarks of the IBM Corporation in the United States and/or other countries:

IBM PS/2
OS/2

The following items used in this publication denoted by a double asterisk (**), are registered trademarks of the following companies.

DEC	Digital Equipment Corporation
INTEL	INTEL Corporation
XEROX	XEROX Corporation
Microsoft	Microsoft Corporation
Novell	Novell Incorporated
Unix	AT&T Corporation
3Com	3Com Corporation

First Edition (January 1991)

Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM* systems, consult the latest *IBM System/370 Bibliography of Industry Systems and Application Programs*, GC20-0370, for the editions that are applicable and current.

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 program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

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 Commercial relations, IBM Corporation, Purchase, NY 10557.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department E02, PO Box 12195, Research Triangle Park, North Carolina, U.S.A. 27709. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This manual describes the architecture of Heterogeneous Local Area Management.

Who Should Read This Book

This book is for programmers who wish to implement Heterogeneous LAN Management.

Corequisite Publications.

IBM/3Com Heterogeneous LAN Management Technical Reference, SC30-3540.

Related Publications

External IBM publications

- *IBM Systems Network Architecture Management Services Reference*, SC30-3346.
- *IBM Token-Ring Network Architecture Reference*, SC30-3374-02

OSI Standards References

- ISO/TC 97/SC 21N 1373, *Common management Information Service Definition*
- ISO/TC 97/SC 21N 1375, *Common management Information Protocol Definition*
- ISO/IEC 7498, *Information processing systems - Open Systems Interconnection - Basic Reference Model.*
- ISO/IEC 7498-4, *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 4.*
- ISO/IEC 8824, *Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).*
- ISO/IEC 8825, *Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*
- ISO/IEC 9072-1, *Information processing systems - Open Systems Interconnection - Remote Operations - Part 1: Model, Notation and Service Definition.*
- ISO/IEC 9072-2, *Information processing systems - Open Systems Interconnection - Remote Operations - Part 2: Protocol Specification.*
- ISO/IEC 9595, *Information processing systems - Open Systems Interconnection - Common Management Information Service Definition.*
- ISO/IEC 9596, *Information processing systems - Open Systems Interconnection - Common Management Information Protocol Specification.*
- ISO/IEC 9596, *Information processing systems - Open Systems Interconnection - Common Management Information Protocol Specification.*
- ISO/IEC 2nd DP10040, *Information processing systems - Open Systems Interconnection - Systems Management Overview.*

- ISO/IEC 2nd DP10040. *Information processing systems - Open Systems Interconnection - Systems Management Overview.*
- ISO/IEC DP 10164-5, *Information processing systems - Open Systems Interconnection - Systems-Management - Part 5: Event Report Management Function.*
- ISO/IEC DP 10165-4, *Information processing systems - Open Systems Interconnection - Management Information Services - Structure of Management Information Part 4: Guidelines for the Defintiions of Managed Objects.*

Contents

Figures

Tables

Chapter 1. Heterogeneous LAN Management Overview

1.1 Problem Statement

With the advent of desktop computing and the personal workstation, networks have been radically transformed. What was once a centralized, "Glass House," enterprise has now been expanded to include distributed processing power on the desktop. With this distribution of processing power, encouraged by the advent of the Local Area Network, has come new problems.

Network management, often overlooked, but mandatory, was once a function that could be performed centrally on those few components, CPUs and controllers, that made up the connection sub-system. In the world of the Local Area Network (LAN), it has taken on new meaning. LANs distribute the connection mechanism, making the question of management a more difficult one. We are now forced to manage more than just a small group of local machines. We must now manage all those distributed devices that make up our new network.

With the expanded scope and flexibility in connectivity brought by the Local Area Network, it is not uncommon to see a LAN supporting several "Logical" functions. That is, LANs are not, by their very nature, limited to the support of a single function, but rather provide a shared connection sub-system. In large installations, the common LAN may be providing host connectivity for some users, while accommodating several LAN File and Print servers for others. It is therefore important to view the LAN as a connection provider for many, and varied uses.

To further complicate the problem, there exist multiple distinct LAN topologies, and in practice it is common to see these varied topologies existing within a single network. Therefore, it is important that any management solution be media independent.

In order to address the management needs of the Local Area Network environment, it is important to consider the following points. First, LANs enable a dispersed environment. Secondly, the connectivity provided by the LAN is not limited to a single use, so management of the system should not be tied to any one use. Thirdly, LANs exist in varied types, so management can not be tied to a single topology. Lastly, to meet these goals, and to increase the acceptance level by the industry, management protocols must be based on existing standards, where applicable.

1.2 Alternatives

Before it is possible to examine some of the considered alternatives, it is important to possess an understanding of the management model. Distributed management necessitates the existence of two basic entities, a centralized collection-control point, or points, and an agent in all managed devices. In this model, the controlling entity is capable of making request of the agent, and the agent is capable of sending data to the controller. This could, and often is thought of as a type of database, where the managed devices act as a distributed repository of the data, and the controlling points, as users of that data. Based on this model, it is then apparent that three things are needed: First, a common methodology for repres-

enting the data that these devices store, a common MIB (Managed Information Base). Secondly, common formats and protocols must be used to communicate the management functions. Lastly, a common transport protocol must be used to move the management protocol through the network. Once all three of these items are met, then distributed management can exist. To address all of the requirements that have been stated, several alternatives were examined, and for various reasons, rejected.

1.2.1 Completely Proprietary Protocol:

The alternative was rejected based on the early premise that any definition should be based on existing standards where applicable. The work being done by the International Standards Organization (ISO) to define Common Management Information Protocol (CMIP)/Remote Operation (RO) provided a well defined solution for the flow of management information that satisfied the requirements for LAN Station Manager.

1.2.2 CMIP/RO Over full stack (seven Layers), Using IEEE 802 Defined

Objects:

While this alternative is most likely to conform to the full ISO standard for CMIP/RO in the future, the required overhead to emulate a full stack was perceived as prohibitive. Additionally, at the current time, there are no standards for the layer managed object definitions, and the Structured Management Information (SMI) standards are not mature enough for use in this architecture.

1.2.3 CMIP/RO Over Logical Link Control (LLC) Type 1, Using IEEE 802

Defined Objects:

This solution too was rejected because as stated above, the layer managed object definitions are not mature enough for use in this architecture.

1.3 Proposed Solution

The solution that is outlined in this architecture represents an implementation of CMIP/RO over Logical Link Control (LLC) type 1, using proprietary definitions for managed objects. CMIP/RO is used because it is an existing and well defined architecture, and because it holds the promise of wide spread standards acceptance. LLC type 1 (based on direction of the standard) is used because it is the standards based approach to handling multi-protocol implementations. Proprietary object definitions are used because they appear to be closest to the direction of future standards.

1.4 Local Area Network (LAN) Station Manager Description

The LAN Station Manager, as described by this document, is an application designed to enable network management across mixed media Local Area Networks. This application may reside in any LAN attached device, and allow that device to either be managed, or by virtue of a management application designed to make use of LAN Station Manager, to manage network devices. This description is intended to be not only network media independent, but also independent of operating systems, hardware platforms, and user applications, resulting in a definition that is a single common platform for network management.

1.4.1 Functions enabled/provided by LAN Station Manager:

- Maintains and reports pertinent counter and configuration information about the LAN attached station in which it resides.
- Enables remote management applications to Get/Set configuration information and counter threshold values.
- Releases unsolicited event reports to registered remote management applications.
- Provides common protocols, transport, and connection mechanism for the attachment of both additional Managed entities, and Managing Process applications.

1.4.2 Scope of This Document:

This document will define all of the formats, protocols, services, and interfaces required by the LAN Station Manager. This includes the International Standards Organization (ISO) CMIP/RO protocols, and the required support structures to accommodate operation at the LLC layer, as well as a breakdown of the required commands and their formats. It will further detail the structure of all the exposed interfaces, and the LLC name management services provided, and used by the Station Manager. Finally, this document will detail the definitions for managed objects for the Token-Ring, Ethernet, and PC-Network environments.

Chapter 2. Statement of Direction

2.1 Standards

Portions of this architecture have been presented to and accepted by the IEEE 802.1 standards body. Further, it is understood that when the SMI and layer management object standards have been approved, this architecture may have to restructure its data to conform to these new standards. It is also understood that the object definitions contained in this document may, as a result of submission, change to reflect the consensus of the standards body.

2.2 Future Implementations

It is the intent of this architecture to be completely independent of media, operating system, hardware platforms, and user applications. With this as a premise, future implementations are not bounded by the formats and protocols of this architecture, but rather by the definitions of managed objects required for that implementation. This assumes that the new media is capable of supporting the required transport, LLC type I. It should be noted, that this transport is an industry standard, and its implementation on new media is anticipated. As an example, the currently emerging technology of FDDI includes a definition for an LLC transport.

Chapter 3. Heterogeneous LAN Management Structure

3.1 Management Model

3.1.1 Architectural Intent

The design intent behind this architecture is to create a platform for management services. That platform should perform the following functions:

- Provide external communications, using standards based management protocols
- Provide APIs for Managed Entities, that is functions wishing to be managed
- Provide APIs for "Managing Process," controlling functions.

That in order to accomplish this, functions must be defined for the gathering of information, and for controlling the resources that collect this information. This architecture allows for the maintenance of counters and other pertinent information, where maintenance implies that these "attributes" can be set and retrieved by a remote managing process, and in the instance of specialized attributes, like counters, they are capable of emitting unsolicited event notifications.

All management that flows between stations, is accomplished using RO/CMIP, transported over LLC type I services. These flows are used for getting and setting parameters, issuing notifications, and performing selected actions.

3.1.2 Model Considerations.

The ISO CMIP/Common Management Information Service (CMIS) model was used as a basis for the work done in this document. However, several concessions were made to facilitate implementation of this model on the widest possible platform base. While the ISO Model relies on the full ISO seven layer stack to provide all the required management services, it was determined that on the platform that would be most prevalent in our initial environment, a PC - PS/2 running DOS, this was not practical due to memory requirements. As a result, a "short stack" version of the ISO model was adopted. This "short stack" version provides the support for RO/CMIP, but at layer 2 in the stack, thus greatly reducing the required overhead on the station. Unfortunately, it also necessitated adding some function to the architecture to replace several of the services usually provided by the higher layers, and forced some service requirements out to the functions being managed. All this was done in a effort to create the best compromise between the standards model and practical, implementation driven reality.

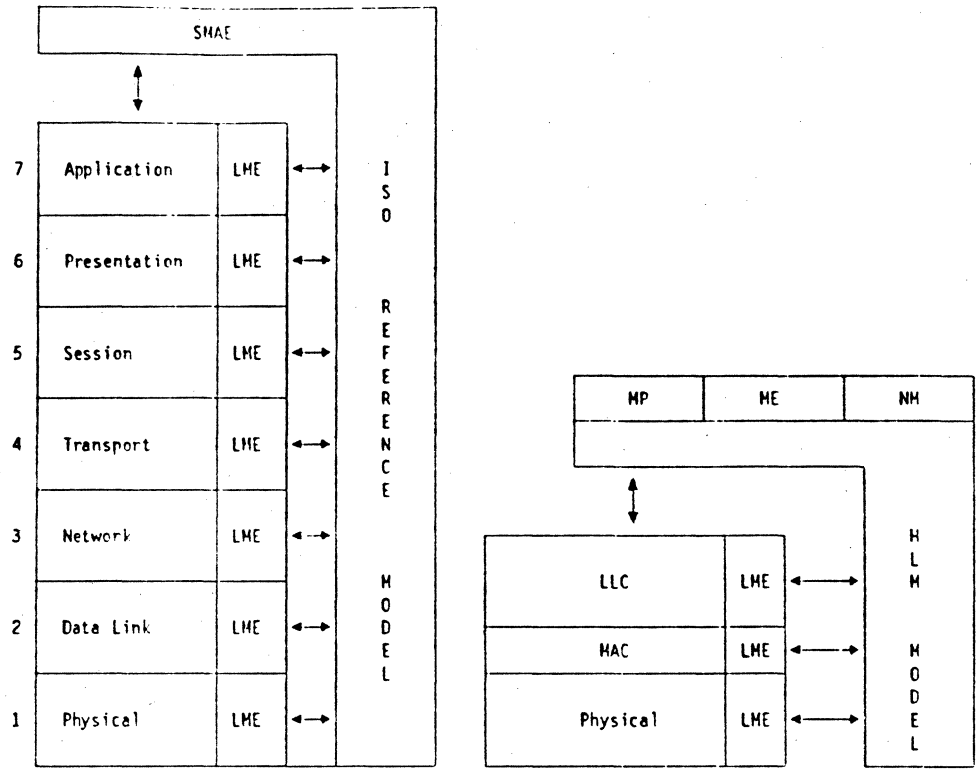


Figure 3-1. Comparison of the ISO 7 Layer Reference Model and the HLM "Short Stack" model

3.1.3 HLM Model.

In the HLM model, it is necessary to collapse some of the function required to accomplish the management functions. For example, name management, National Language support, and function registration, are all functions that should, in accordance with the ISO model reside in one of the higher layers, but in this model, these functions have been moved into the Station manager base function. Another example of function redistribution, is that of the Encoder/Decoder. This function, defined by ISO to be handled by the the SMAE (System Management Application Entity), has been distributed to the attached LMEs (Layer Management Entities), and MEs (Managed Entities), in an effort to ease memory requirements on certain implementations.

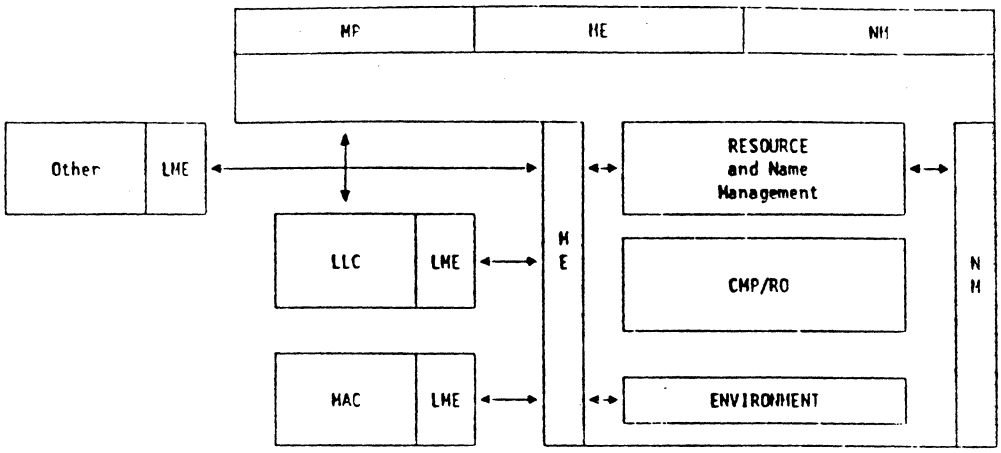


Figure 3-2. HLM Reference Model

The Layer Management Entity (LME) is a component of the system that is responsible for providing management information to the station manager, on behalf of a layer of the protocol stack, such as a Token Ring MAC layer. In the HLM reference model, as pictured in Figure 3-2, the interface for the LME has been retained. However, as a result of the "short stack" design requirement, the interface is also exposed as the ME, or Managed Entity, API. In this way, applications, or other functions can attach to the station manager to make use of its management services, and transport. This adds the ability to separate the ME interface if required, as in the case of OS/2*, where it may run in a context other than that of a device driver. The interface is essentially a CMIP interface, that is, the information that flows across it is the CMIP PDU. Both LMEs and MEs are required to service all incoming CMIP commands, and to issue the appropriate CMIP notifications. Attached MEs are treated, by the station manager, in exactly the same manner as attached LMEs.

The LLC Name Management (LNM) interface is intended to provide the name services required for the LLC transport. Figure Figure 3-2 indicates that, like the LME/ME interface, the LNM/NM interface can also be split to support implementation requirements.

To overcome the absent layer 7 CMIS services as defined in the ISO SMAE model, HLM exposes a Managing Process (MP) interface. This interface is essentially an RO (Remote Operations) CMIP interface. This allows applications that write to this interface to issue RO/CMIP invocations to the station manager, and in turn to the appropriate devices on the network. It further enables the collection of unsolicited CMIP notifications.

Having examined the external interfaces to station manager, there are a few internal components that should be discussed. The Environment Class; this is essentially an LME for the station. Station specific information that should vary, depending on the type of station. For an example, see the Environment Object Class definition in Chapter 20, this defines the environment class for a PC - PS/2 workstation. Resource and Name Management; These are also LMEs that provide access to further station information, such as the station object containment, and the list of registered names. For further details, see Chapter 20.

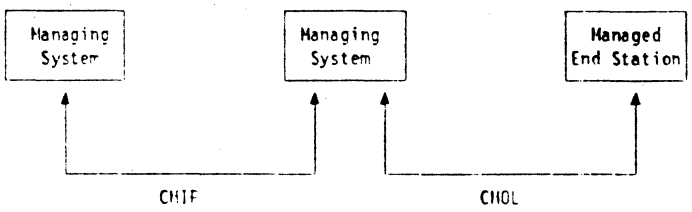
3.1.4 Station Relationships.

It is important to consider the HLM model as an enabler/provider of system management function. While the base station manager provides a certain amount of management function, as described above, its primary role is that of an enabler for attaching functions. These functions can be either ME (Managed Entity) or MP (Managing Process) in nature, but they derive their true function only through the station manager. It should be noted, that there exists no hierarchy, that all network attached stations, with a station manager active, are peers. That "control," or management function exists only through the MP interface of station manager.

3.2 Protocol Boundaries and HLM Positioning

By design, HLM is architected to provide a management solution for a broad scope of end user devices configured on MAC bridge connected LANs of differing media types. The protocol of choice, (CMOL) provides an assurance of inter-operability at the DLC level across the broadest set of possible architectures and media types. The managing system-to-managed-entity focus fits nicely with the OSI management model of use of full CMIP between managing_system and CMOx for managing_system-to managed-entity_services. As defined, CMOL is architected to not only provide an assurance of inter-operability between systems, but be implemented across all of the popular end-station environments, including the memory constrained MS-DOS systems.

CMIS (end to end)



Chapter 4. Lower-layer Services Architecture (LSA) for LAN Station Manager

This section describes the Lower Layer Services architecture for Local Area Network Station Manager. The interfaces defined in this architecture support Name Management and Resource Management.

This section assumes that the reader has a working knowledge of Remote Operations (as described in ISO documents 9072-1 and 9072-2), Common Management Information Protocol (as described in ISO 9596), Common Management Information Service definition (as described in ISO 9595), and Abstract Syntax Notation (ASN.1 as described in ISO 8824 and ISO 8825.)

This section does not describe any specific IBM product (machine or program) or service. Neither should information herein be construed to mean that all or any specific IBM products necessarily provide only, or all of, the LAN LLC interface functions described herein. Refer to appropriate IBM product description and operation manuals for information regarding the availability and characteristics of LAN LLC interface functions supported by specific IBM products.

A LAN Station Manager (LANSM) is a collection of functions that perform Name Management and Resource Management. LSA provides an architected definition of service interfaces that support Name and Resource Management.

The Name management LSA interface provides a LANSM user a mechanism for performing address resolution and route discovery

The Resource Management LSA interface provides a mechanism for creating, maintaining and reporting Managed Objects. Resource Management is composed of two LSA interfaces: Managing Process and Managed Entity. The Managing Process interface allows a LANSM user to manage objects using Remote Operations. The Managed Entity interface allows the LAN Station Manager to perform operations on objects owned by local LME's.

4.1 Lower-Layer Services Architecture Synopsis

The Lower-Layer Services Architecture (LSA) defines a single, consistent architecture comprised of the concepts, terminologies, and structural framework with service definitions for the lower three layers of the OSI Reference Model.

Generally, LSA provides:

- A broad range of communications support
  - Provides a framework flexible enough to accommodate a variety of existing and evolving communications protocols, as well as being open-ended enough to allow for additional service definitions to be added as needed.
  - Supports both SNA and non-SNA environments (with the focus on SNA and OSI).
- Open System support

  The services are sub-settable, providing

- Direct access to selected service layers, and
- The capability to add or replace selected service layers in a modular manner.

4.1.1 Scope

The LSA is comprised of the concepts, terminologies, and structural framework with service definitions. It is concerned with the **service definition** and not the protocol specification for the selected service layers:

- The service definition of a given service layer specifies the **"open"** interface through which service users access the specific type/level of services provided by that layer.
- The protocol specifications are contained in protocol architecture Volumes containing specifications as provided by the established standards bodies; any references to the protocol operations are limited to the management and policy aspects of the service definitions.

This architecture defines the service interfaces for the communications functions contained in the lower three layers of the OSI reference model or the "DLC Layer" of the SNA model.

LSA provides a clear and specific context for the communications concepts and their structural relationships as abstract data types. It provides a reference point for a valid implementation of the communications I/O service interface and a stable base of the communications I/O service definitions.

The service definitions of the LSA are independent of the system facilities of the product that will implement the communications services.

4.1.2 Dependencies on Other Architectures

LSA is dependent on the following other supporting architecture functions:

- Operational Environment

  These functions provide code loading and support of inter-layer transport establishment.

- Inter-Layer Transport (ILT)

  These functions provide an entity to entity (or process to process) transport mechanism (connection) which carries the LSA primitives.

4.2 Terms

NEI	Network Entity Identifier
CM	Connection Manager
CNM	Communication Network Management
EE	Entity Enabler
LANSM	LAN Station Manager
LME	Layer Management Entity
LF	Largest Frame
DU	Data User

HB Heartbeat

NEI Database This data base, which is contained in LANSM, is a collection of NEI names and other related information. LSA does not define this data base.

MEA Managed Entity Application - Is an LLC User (e.g., Bridge Server Application, File Server Application, etc.) function that may be present in a station. The LAN Station Manager manages object owned by the MEA via a CNM SAP.

LSA Lower-layer Services Architecture.

SAP Service Access Point.

4.3 LSA LAN Station Manager Reference Model

This is an abstract model of a real open system. As a conceptual and functional framework, its goal is to define a common basis for coordinated development.

The LSA LAN Reference Model consists of the following functional entities and external resources.

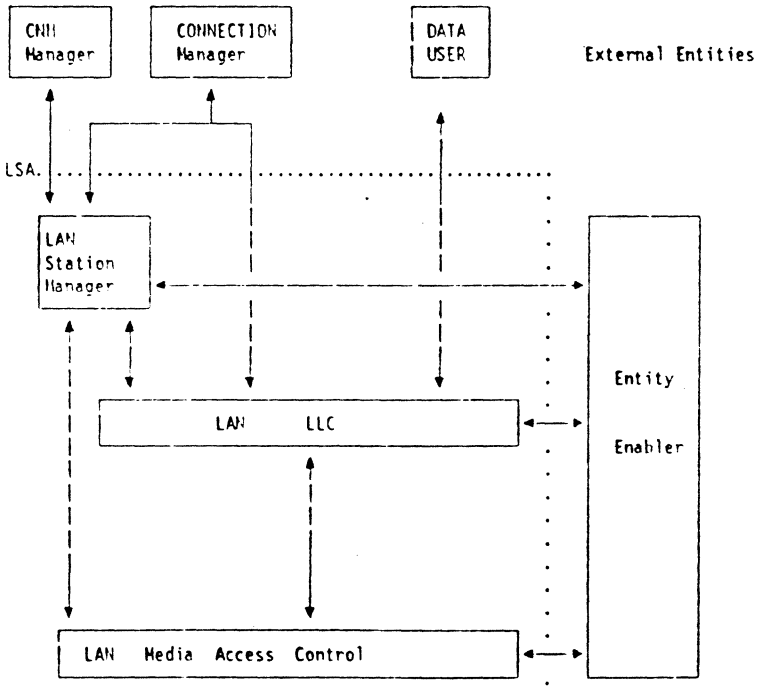


Figure 4-1. LAN Station Manager, LLC, and MAC LSA Model

4.3.1 Required External Resources

The upper level communication software can be of various flavors, SNA, OSI, etc. Because we can not assume any one particular architectural configuration of the upper level communication support, we have chosen instead to identify FUNCTIONS that must be performed by the upper level software. These functions are depicted in the figure by the CNM, CONNECTION, the ENTITY ENABLER, and the DATA USER.

These external resources are needed to define where LSA sends asynchronous primitives.

Note that all of the external resources may not be required by every protocol. The LSA Reference Model identifies a superset of external functional resources and each protocol will identify which of them that it requires.

The intent of the LSA "EXTERNAL" resources is to identify all function support that each LSA entity will require from outside of its scope.

These "EXTERNAL" resources are identified to each LSA entity by activating a SAP to the LSA entity and identifying what services are available through this SAP.

CNM Manager

This manager provides Communication Network Management (CNM) functions to LSA. These functions include reporting statistics and counters, setting thresholds, and obtaining and changing configuration information.

Connection Manager

This manager performs the required connection procedures using the primitives defined and supported by LSA. This manager "understands" the physical configuration, causes the proper protocol stack to be built and then issues the correct set of LSA primitives to cause a connection to be established.

The Connection Manager understands LSA resources and is informed if a resource fails.

Entity Enabler

The Entity Enabler component provides the initial configuration parameters to a newly created entity. The Entity Enabler also clears the configuration when desired. The Entity Enabler works in conjunction with the local operating system services which creates the LSA entities and establishes the underlying transport or logical connection mechanism between LSA entities and External resources.

Data User

The Data User uses LSA primitives to send and receive PDUs. In an SNA Node, the Data User is Path Control. .

4.4 LSA LANSM Function Placement

LSA defines the LANSM as an Entity that provides both Name Management and Resource Management functions to the users (external LSA entities) and uses services provided by the LLC, MAC, and Managed Entity Application provider entity instances.

The Managed Entity interface (not explicitly shown) composed of the LAN Station Manager user accessing an LME via an LSA SAP (as a CNM mgr.)

The LAN Station Manager also utilizes connectionless data transport services provided by the LAN LLC to send/receive connectionless data for Name and Resource Management. The LANSM access the LLC via two LSA Data User SAPs (the N refers to Name Management and the R refers to Resource Management.)

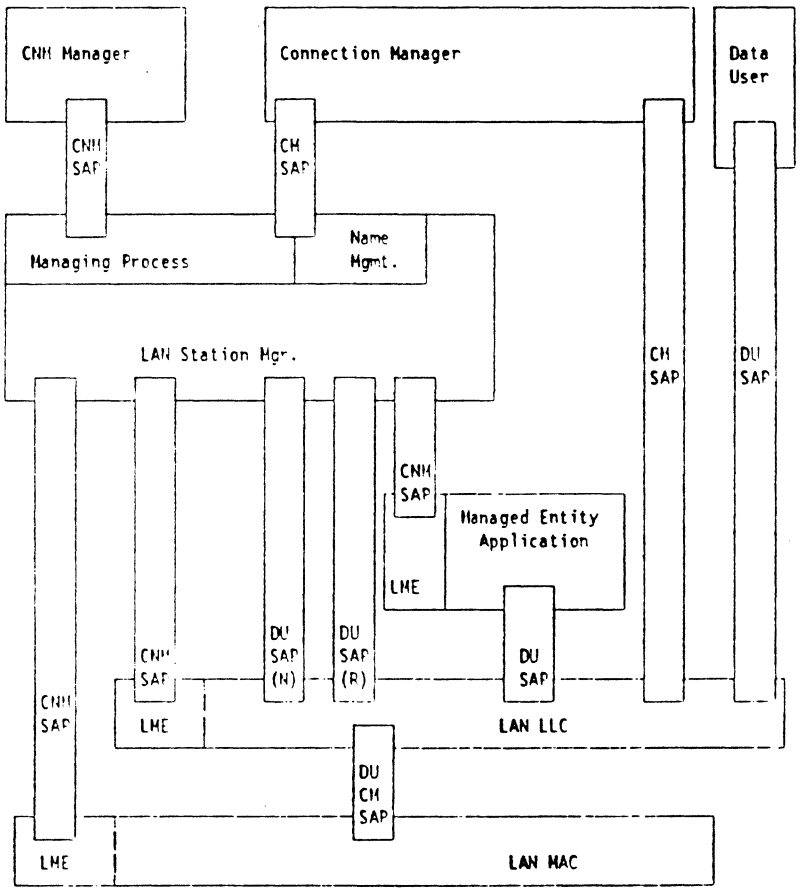


Figure 4-2. LSA LAN Station Manager Function Placement

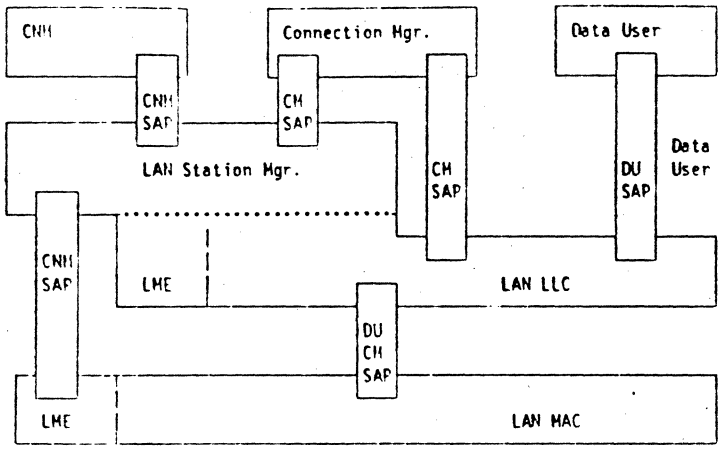


Figure 4-3. LSA LANSM Function Placement with Collapsed LLC LANSM Interface. LSA allows implementations to "collapse" two entities together thus making the interface between the two implementations specific. In this example, the LANSM and the LLC are collapsed, thus the LANSM/LLC interface (denoted by the dots) is implementation specific.

4.5 Resource Management

LSA for LAN Station Manager defines two interfaces involved with Resource Management: Managing Process (MP) and Layer Management Entity (LME).

The Managing Process interface allows a User entity (e.g., a CNM Manager) to rely on the LANSM (provider) for Remote Operations (RO) services. In the following figure, the CNM Manger contains Remote Operations state machine function. The CNM Manager relies on the LANSM to build the RO frame based on the primitive (INVOKE, REJECT, ERROR, RESULT) and to guarantee delivery (via retries, etc).

The Invoke instance is a construct that is presumed to exist in the LANSM. It is created upon receipt of a INVOKE.request primitive and contains information (e.g.,INVOKE ID) necessary to associate incoming frames (REJECT, ERROR, RESULT) with the original INVOKE request.

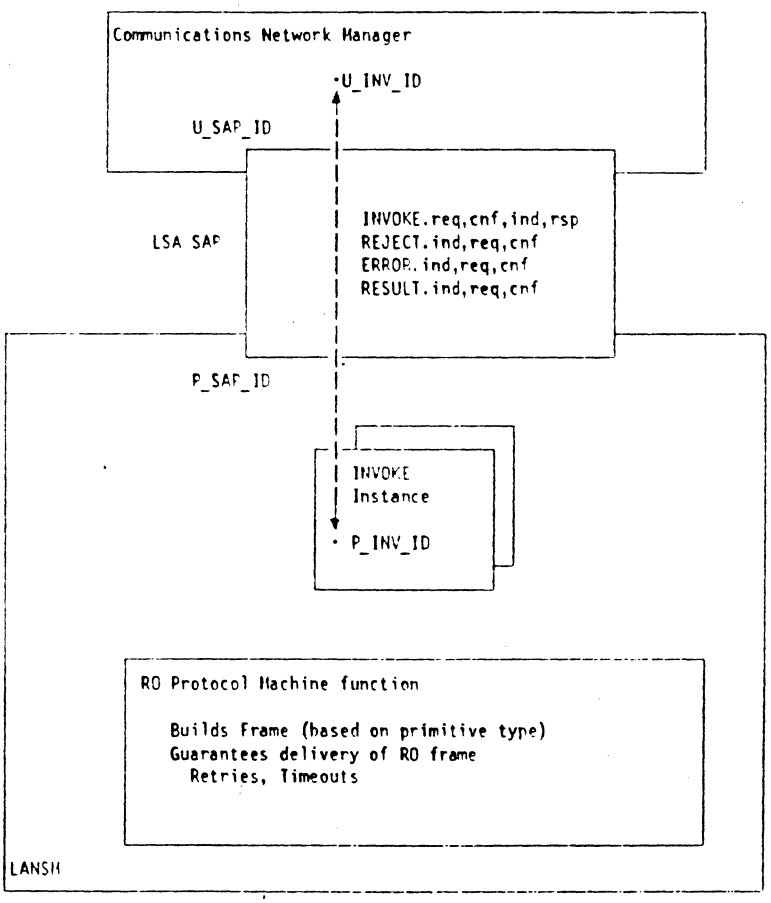
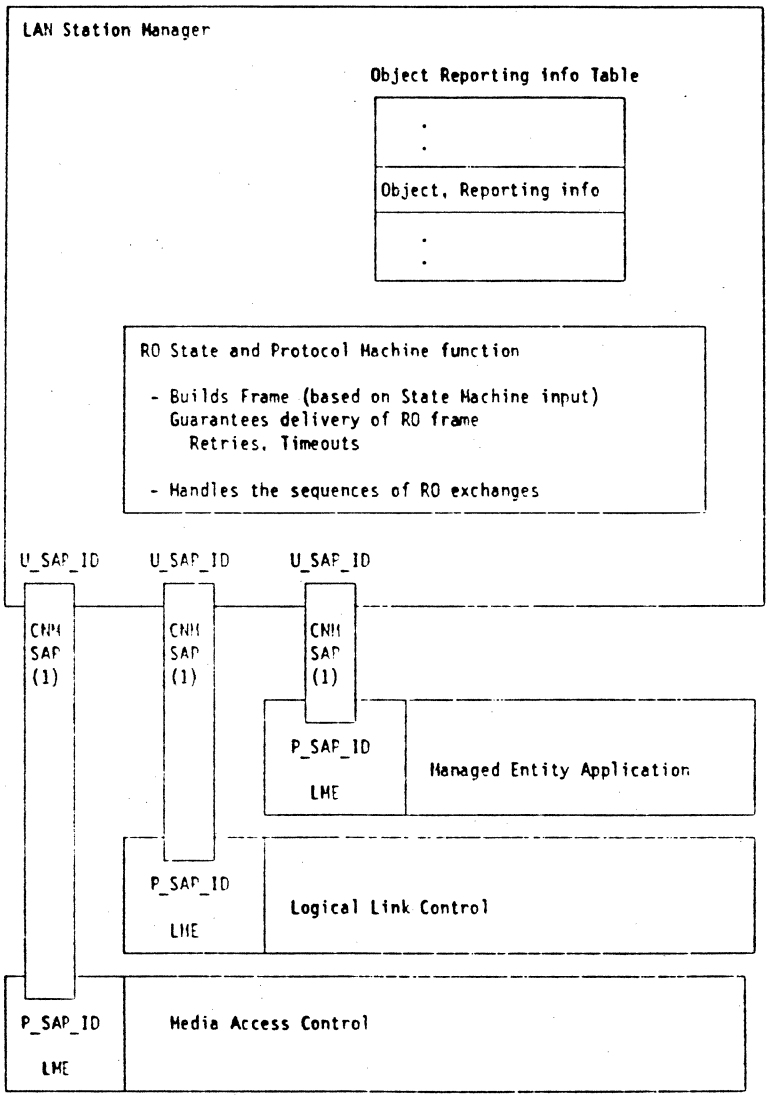


Figure 4-4. Managing Process interface of LAN Station Manager

The Managed Entity (LME) interface is allows the LANSM (user) to perform CMIP/RO functions and to perform local operations (GET, SET, EVENT, ACTION, CREATE, DELETE) on local provider LMEs.

In the following figure, the LANSM is assumed to have the capability to guarantee delivery of RO frames and also have RO state machine function. The Object Reporting info Table (which is a construct that is presumed to exist in the LANSM) contains information on where (e.g., address of a remote manager) the LANSM reports EVENTS (depending on the objects contained in the EVENT).



- The LSA primitives that may flow through this SAP are GET.req/cnf, SET.req/cnf, EVENT.ind, ACTION.req/cnf, CREATE.req/cnf, DELETE.req/cnf.

Figure 4-5. Layer Management Entity interface of LAN Station Manager

4.5.1 Resource Management Flows

The following example flows show LANSM resource management functions performed by LSA primitives.

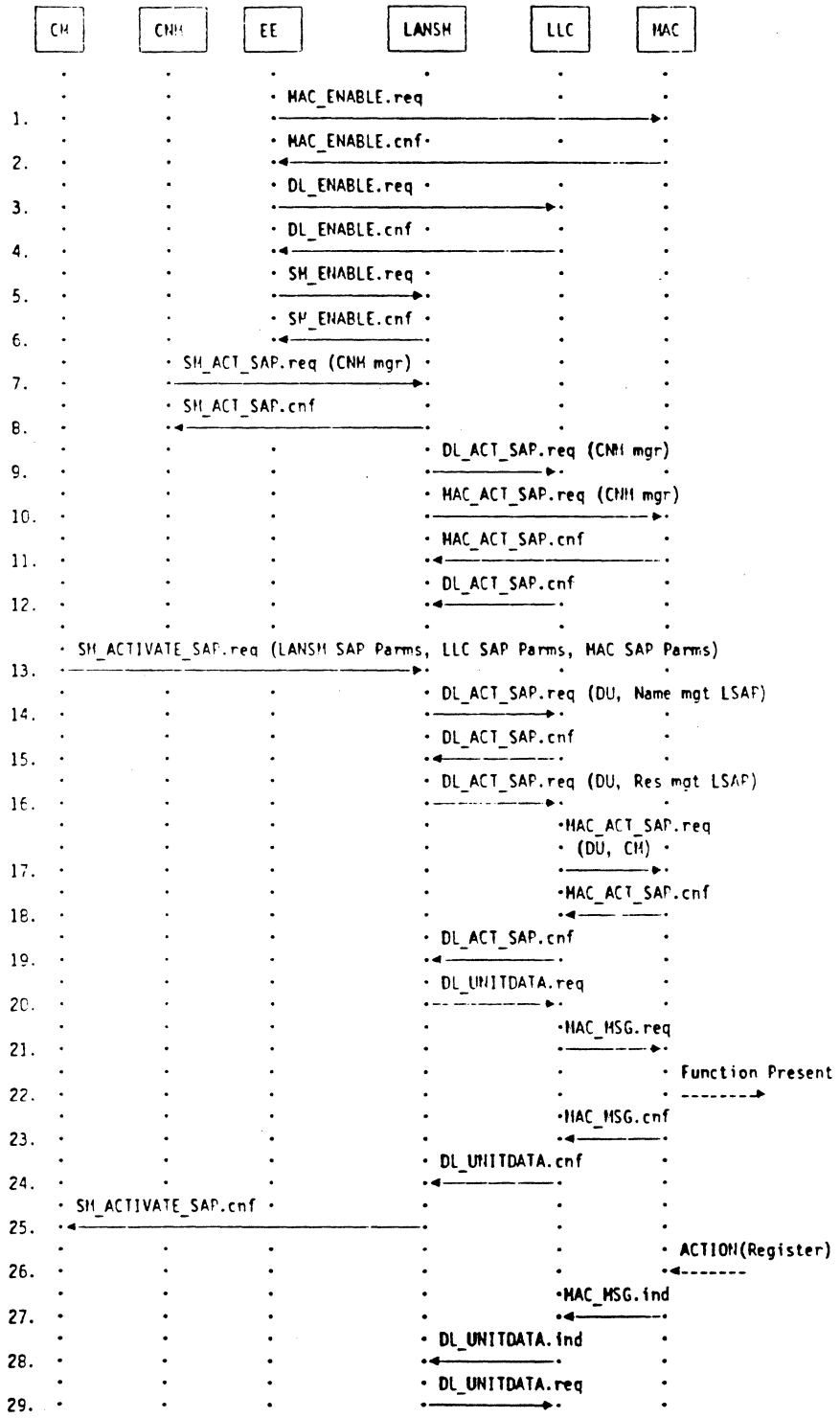


Figure 4-6. Registration Flow

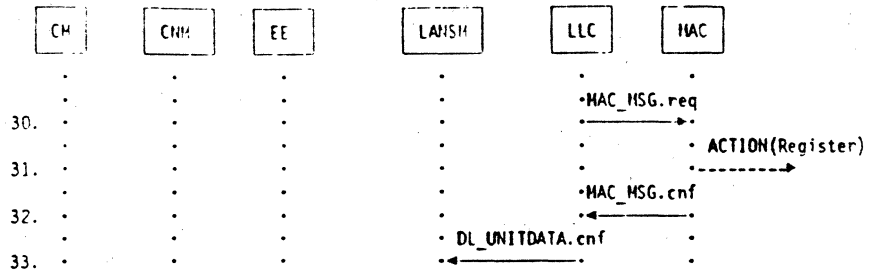


Figure 4-7. Registration Flow

The following notes are the descriptions of the numbered primitive flows shown above.

1-6 The Entity Enabler passes configuration parameters to the MAC, LLC and LANSM entities.

7 The CNM Manager activates a SAP to the LANSM and identifying itself as the CNM Manager to the LANSM. At this time the LANSM specifies in the Encoding_Rule parameter in the ACTIVATE_SAP primitive which indicates the encoding rule (e.g., ASN.1) used for the CNM data passed in the LSA primitives (e.g., INVOKE, RESULT,....) and the version number of the architecture supported.

8 LANSM confirms that the CNM SAP was activated.

9-12 The LANSM activates a SAP to the LLC LME and identifies itself as the CNM Manager to the LLC. The LANSM also activates a CNM SAP to the MAC LME. At this time the LANSM specifies in the Encoding_Rule parameter in the ACTIVATE_SAP primitive which indicates the encoding rule (e.g., ASN.1) used for the CNM data passed in the LSA primitives (e.g., GET, SET, EVENT,...) and the version number of the architecture supported.

13 The combined Data User and Connection Manager activates a SAP to the LANSM and passes SAP configuration parameters for the LLC and MAC.

14,15 The LANSM activates a Data User SAP (including the Name Mgmt. LSAP value) to the LLC to be used for Name Mgmt. primitive flows.

16 The LANSM activates a Data User SAP (including the Resource Mgmt. LSAP value) to the LLC to be used for Resource Mgmt. primitive flows.

17,18 The LLC activates a SAP (Data User, and Connection Mgr.) to the MAC.

19 The Data User (Resource Mgmt.) SAP is confirmed.

20-22 The LANSM begins the Registration process by composing the connectionless Function Present Event frame and passes it to the LLC (via the Resource Mgmt. SAP), which in turn passes it to the MAC, which sends the frame.

23 The MAC confirms that the frame was sent.

24 The LLC confirms that the data was sent.

25 The LANSM confirms that the SAP was opened.

26-28 The MAC receives an ACTION (Register) frame from the remote station (e.g. LAN Manager) and passes it to the LLC. The LLC passes it to the LANSM via the Resource Management SAP.

29-33 The LANSM interprets the I Field of the frame and responds with the appropriate ACTION (Register Response) frame.

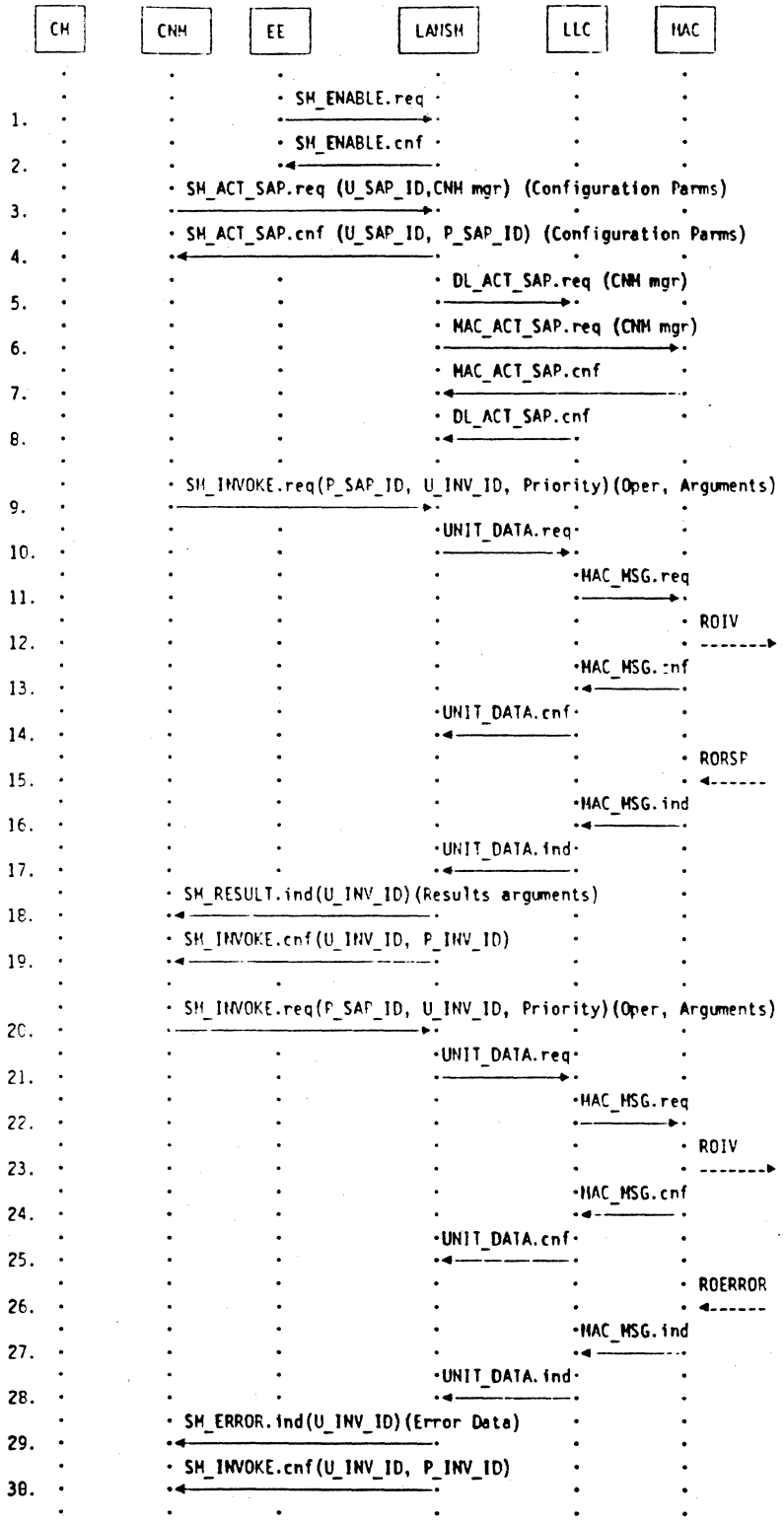


Figure 4-8. Managing Process INVOKE, ERROR, REJECT flows

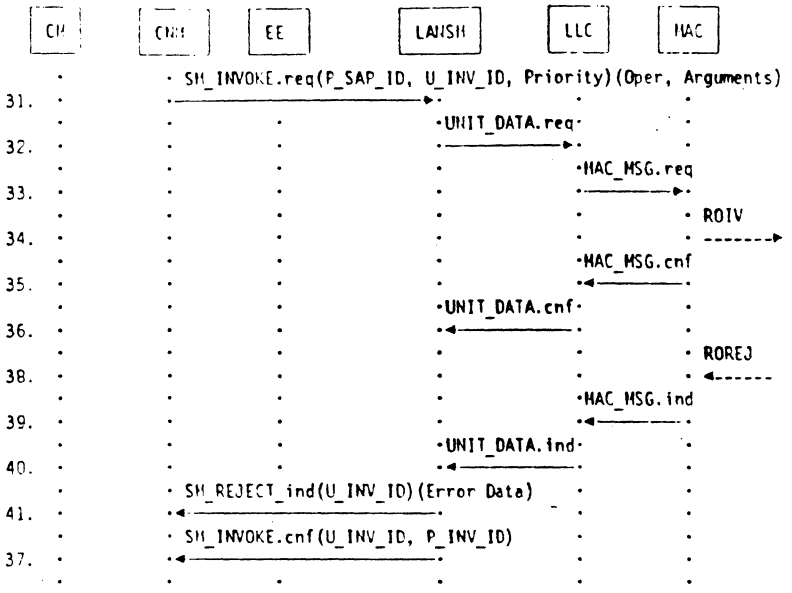


Figure 4-9. Managing Process INVOKE, ERROR, REJECT flows

The following notes are the descriptions of the numbered primitive flows shown above

1,2 The Entity Enabler passes configuration parameters to the LANSM entity.

3 The CNM Manager activates an interface SAP to the LANSM and identifying itself as the CNM Manager to the LANSM. At this time the LANSM specifies in the Encoding_Rule parameter in the ACTIVATE_SAP primitive which indicates the encoding rule (e.g., ASN.1) used for the CNM data passed in the LSA primitives (e.g., INVOKE, RESULT,....) and the version number of the architecture supported.

4 LANSM confirms to the CNM mgr. that the CNM SAP was activated. This also allows configuration parameters to be returned if they are assigned in the LANSM.

5-8 The LANSM activates a SAP to the LLC LME and identifies itself as the CNM Manager to the LLC. The LANSM also activates a CNM SAP to the MAC LME.

9 The CNM mgr. issues an INVOKE primitive to the LANSM. This primitive contains a U_INV_ID (assigned by the CNM mgr), RO Operations and CMIP Arguments, REMOTE SAP/MAC ADDRESS.

10-14 Upon receipt of the INVOKE primitive, the LANSM creates an INVOKE ID, assigns a P_INV_ID, builds a RO-INVOKE frame, and passes it to the LAN LLC where it is sent as connectionless data.

15-17 When the RO-RESULTS frame is received, it is passed to the LANSM.

18 The LANSM compares INVOKE IDs in the frame with the INVOKE IDs in the INVOKE instances and them passes the results to the CNM manager using the appropriate U_INV_ID.

19 The LANSM confirms the INVOKE request was done.

20-25 The CNM manager issues another SM_INVOKE primitive to cause a RO_INVOKE frame to be sent to a remote station.

26-28 A RO-ERROR frame is received from the remote station and is passed to the LANSM.

29 The LANSM compares INVOKE IDs and indicates to the CMN manager that the RO-ERROR was received.

30 The LANSM confirms the INVOKE request was done.

31-41 The CNM manager sends another RO_INVOKE message, this time it is rejected and the LANSM indicates this to the CNM manager.

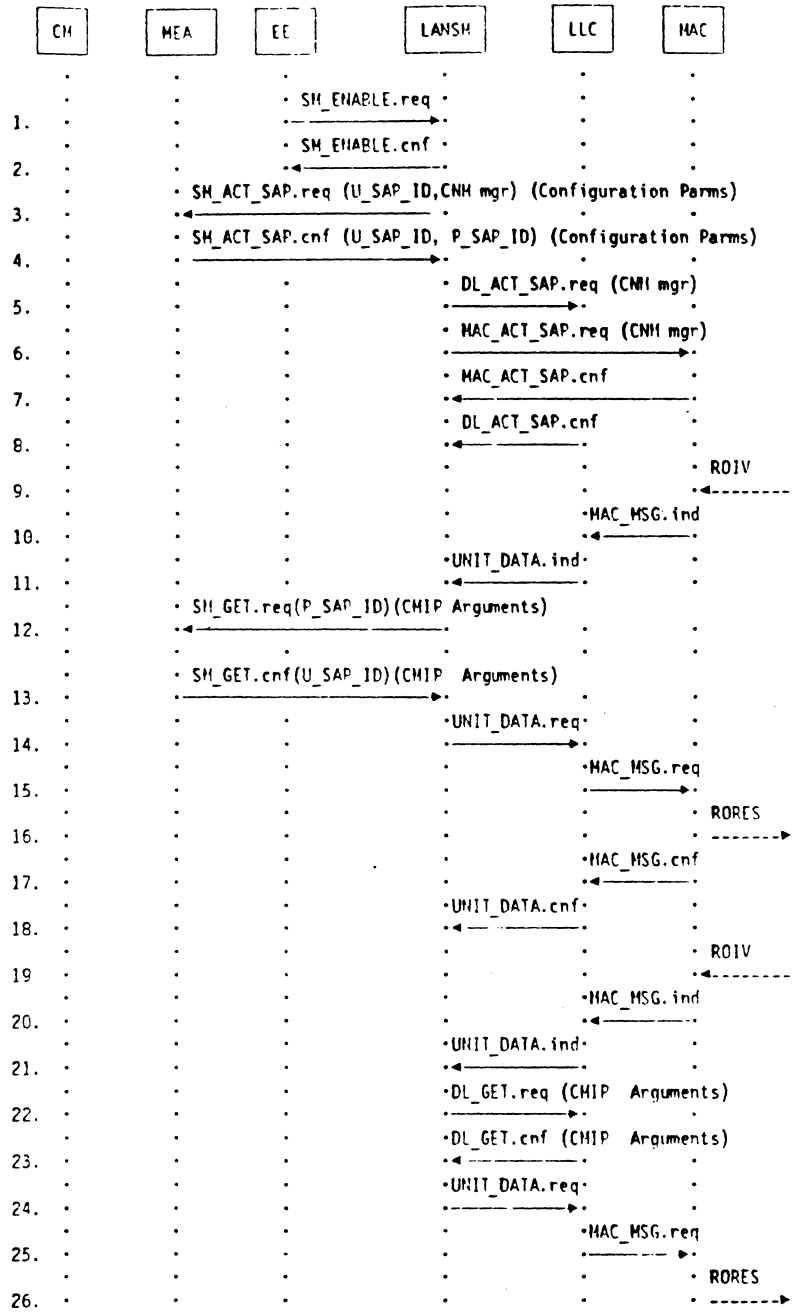


Figure 4-10. Managed Entity flows

1,2 The Entity Enabler passes configuration parameters to the LANSM entity.

3 The LANSM activates an LSA SAP to the Managed Entity Application's LME.

4 The MEA confirms to the MEA that the CNM SAP was activated and passes the Objects (that are "owned" by the MEA LME e.g., Object Class and Object Instance) to the the LAN Station Manager This would allow the LAN Station Manager to associate objects with LMEs.

5-8 The LANSM activates an LSA SAP to the LLC LME and identifies itself as the CNM Manager to the LLC. The LANSM also activates an LSA CNM SAP to the MAC LME. At this time the LANSM specifies in the Encoding_Rule parameter in the ACTIVATE_SAP primitive which indicates the encoding rule (e.g., ASN.1)
used for the CNM data passed in the LSA primitives (e.g., GET, SET, EVENT,...) and the version number of the architecture supported.

9-11 Upon the receipt of a RO-INVOKE frame, it is passed to the LANSM.

12 The LANSM interpret the CMIP arguments and determines that they are managed by the Managed Entity Application's LME (this could be done by matching the objects to a list of objects included in the parameter list when the CNM SAP was activated) and issues a Get.request to the LME.

13 The MEA LME performs the appropriate operation and passes back the results in a GET.cnf.

14-18 The LANSM creates a RO_RESULT's frame and passes it to the LAN LLC where it is sent as connectionless data.

19-21 Another RO-INVOKE is received and is passed to the LANSM.

22-23 The LANSM interprets the CMIP arguments and determines that they refer to objects in the LLC (e.g., by comparing the object class and object instance from the incoming data with object class and object instances associated with LSA SAPs (step 4), the appropriate primitive may be sent to the correct LLC.) The LANSM issues the appropriate LSA primitives that perform the requested function (e.g, GET) to the LLC.

24-26 The LANSM issues a RO-RESULTS from back to the managing station.

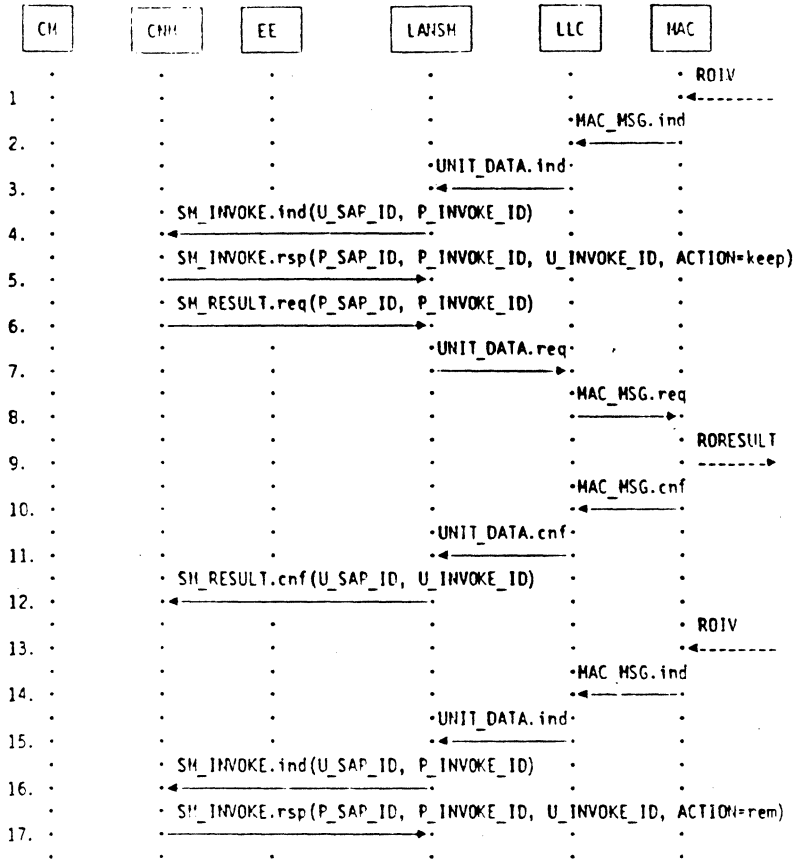


Figure 4-11. Managing Process Interface for receiving an unsolicited INVOKE request.

1,2,3 A remote station sends an unsolicited INVOKE frame (RO-INVOKE-EVENT-REPORT)_and it is passed to the LANSM.

4 The LANSM creates an INVOKE instance and assigns a P_INVOKE_ID, and passes the CMIP data in an INVOKE.ind to the CNM manager.

5 The CNM Manager interprets the CMIP data and determines that the operation is a confirmed operation and issues an INVOKE.rsp (with ACTION=Keep.)

6-11 The CNM manager performs the requested operation, and sends the results (targeted to the invoke instance previously created) to the operation requestor.

12 The LANSM removes the invoke instance and confirms that the result was sent.

13-15 A remote station sends an unsolicited INVOKE frame (RO-INVOKE-EVENT-REPORT)_and it is passed to the LANSM.

16 The LANSM creates an INVOKE instance and assigns a P_INVOKE_ID, and passes the CMIP data in an INVOKE.ind to the CNM manager.

17 The CNM Manager interprets the CMIP data and determines that the operation is a unconfirmed operation and issues an INVOKE.rsp (with ACTION=remove.) The LANSM removes the invoke instance.

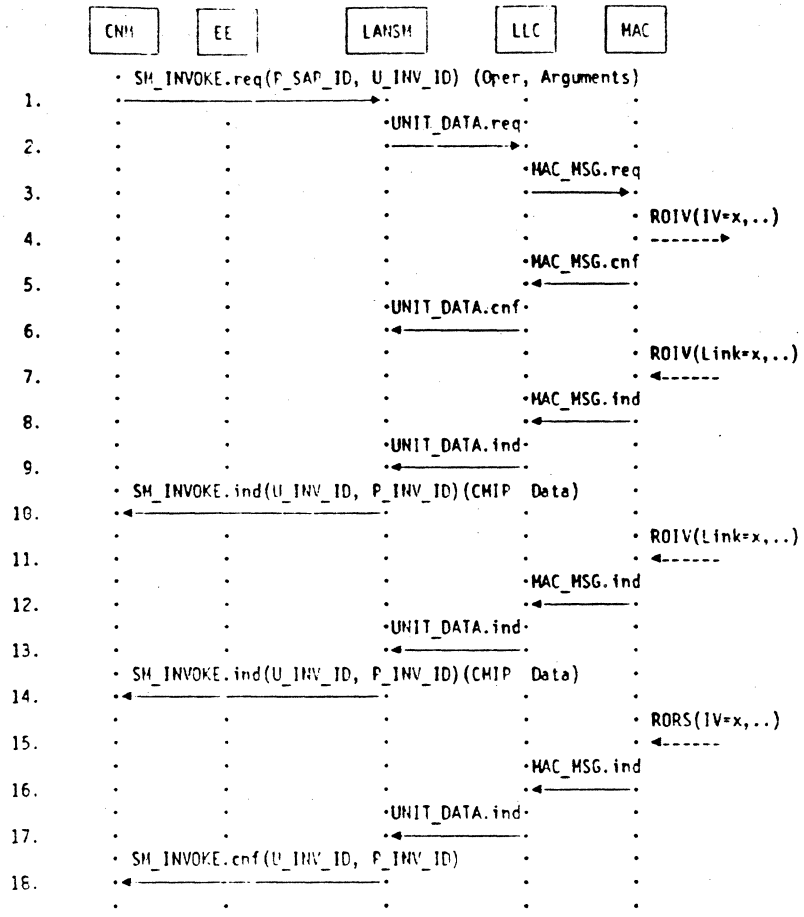


Figure 4-12. Managing Process INVOKE with Linked Replies

The following notes are the descriptions of the numbered primitive flows shown above.

1-6 The LANSM user (CNM manager) sends a INVOKE.req to the LANSM, which causes the LANSM to assign an invoke id (in this case iv=x), form a RO-INVOKE frame and send it.

7-9 A RO-INVOKE is received and passed to the LANSM.

10 The LANSM interprets the frame (determining that it is an invoke and that it is a linked reply) and sends an INVOKE.ind which contains the CMIP data to the LANSM User. This primitive also contains the P_INV_ID of the Invoke instance, which may be required if the CNM mgr needs to send a reply (e.g., RO_ERROR, RO_RESULT,..) prior to the end of the series of linked replies.

11-14 The second linked reply frame is received and passed to the LANSM user.

15-17 An RO_RESULT is received and passed to the LANSM.

18 The LANSM interprets the frame and determines that frame signals the end of the linked replies for invokeid=x and sends an INVOKE.cnf to the LANSM user (CNM mgr)

4.6 Name Management

LSA for LAN Station Manager defines an interface for Name Management which allows a user (e.g., Connection Manager) to perform address resolution and route discovery.

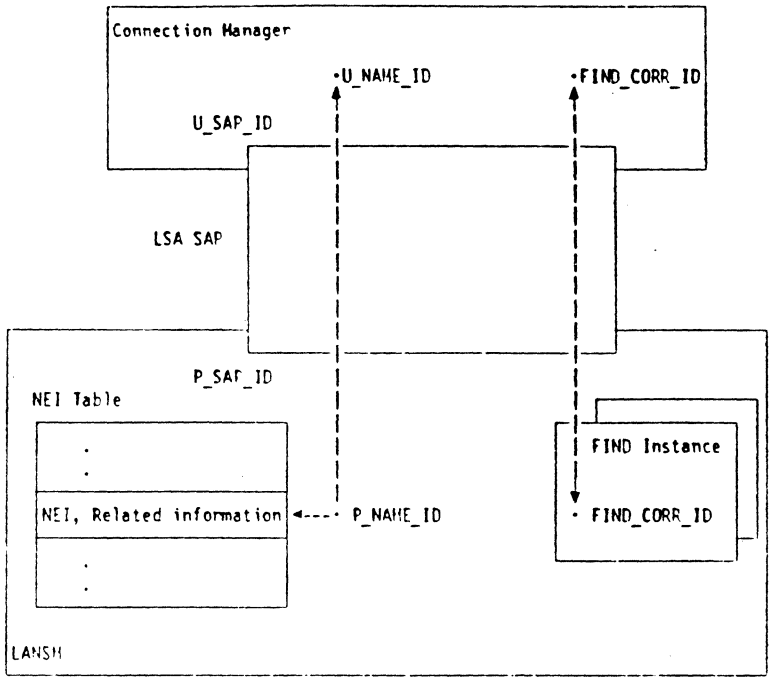


Figure 4-13. LSA LAN Station Manager Identifier relationships. The NEI Table may contain the individual NEIs (and related information) of the users (applications) present in this station. The FIND Instance represent the process of finding a NEI in the Network. The dashed lines indicate the relation ships of User and Provider identifiers.

Note: LSA only defines the LANSM interface, therefore the components like NEI Table and FIND Instance are only examples of constructs or functions that are presumed to exist in LANSM. This also does not preclude any other functions that may exist.

4.6.1 Name Management Flows

The following example flows depict LANSM Name Management functions performed using LSA primitives.

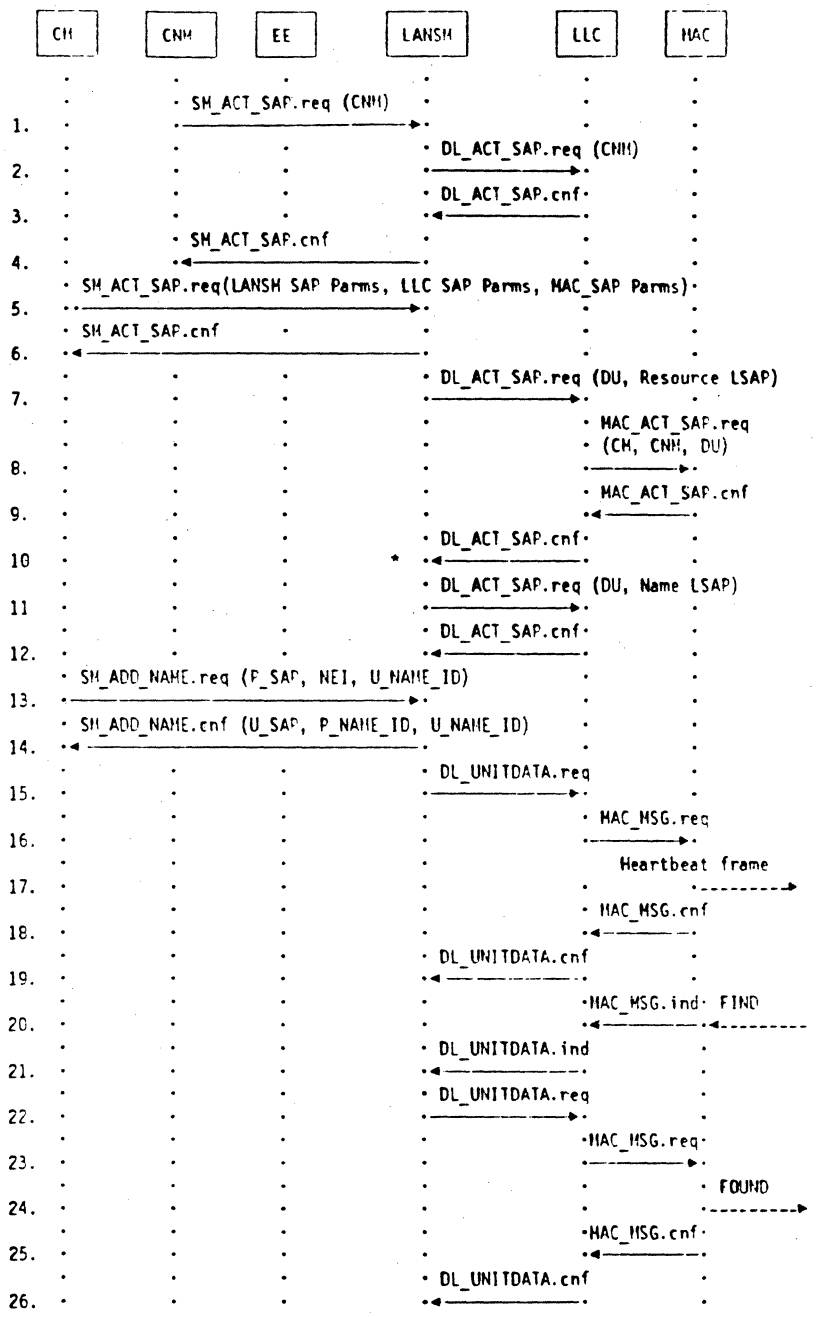


Figure 4-14. Adding a Name flow

The following notes are the descriptions of the numbered primitive flows shown above.

1 The CNM Manager activates a SAP to the LANSM and identifying itself as the CNM Manager to the LANSM.

2,3 The LANSM activates a SAP to the LLC and identifies itself as the CNM Manager to the LLC.

Note: At this time the LANSM may also activate a CNM SAP to the MAC (This would result in separate CNM and CM/DU SAPs).

4 LANSM confirms that the CNM SAP was activated.

5 The Connection Manager activates a SAP to the LANSM and passes SAP configuration parameters for the LLC and MAC.

6 The LANSM confirms that the SAP was activated.

7 The LANSM activates a Data User SAP (including the Resource Mgmt. LSAP value) to the LLC to be used for Resource Mgmt. primitive flows.

8,9 The LLC activates a SAP (Data User, Connection Mgr. and CNM Mgr.) to the MAC. Also, the Locate functional address may be opened by the MAC to allow this station to receive frames from stations in distributed address server mode.

10 The Data User (Resource Mgmt.) SAP is confirmed.

11,12 The LANSM activates a Data User SAP (including the Name Mgmt. LSAP value) to the LLC to be used for Name Mgmt. primitive flows.

13,14 Connection Mgr. adds a Network Entity identifier to the Network Entity identifier database in the LANSM and established a U_NAME, P_NAME relationship. Subsequent request primitives that refer to names in the Network Entity Identifier database will use the P_NAME_ID to refer to a specific name.

15-19 If the ACTION=Heartbeat was present in the ADD_NAME.req then the LANSM Heartbeats for that name.

20-26 A FIND is received from the Network, which results in a FOUND being sent back.

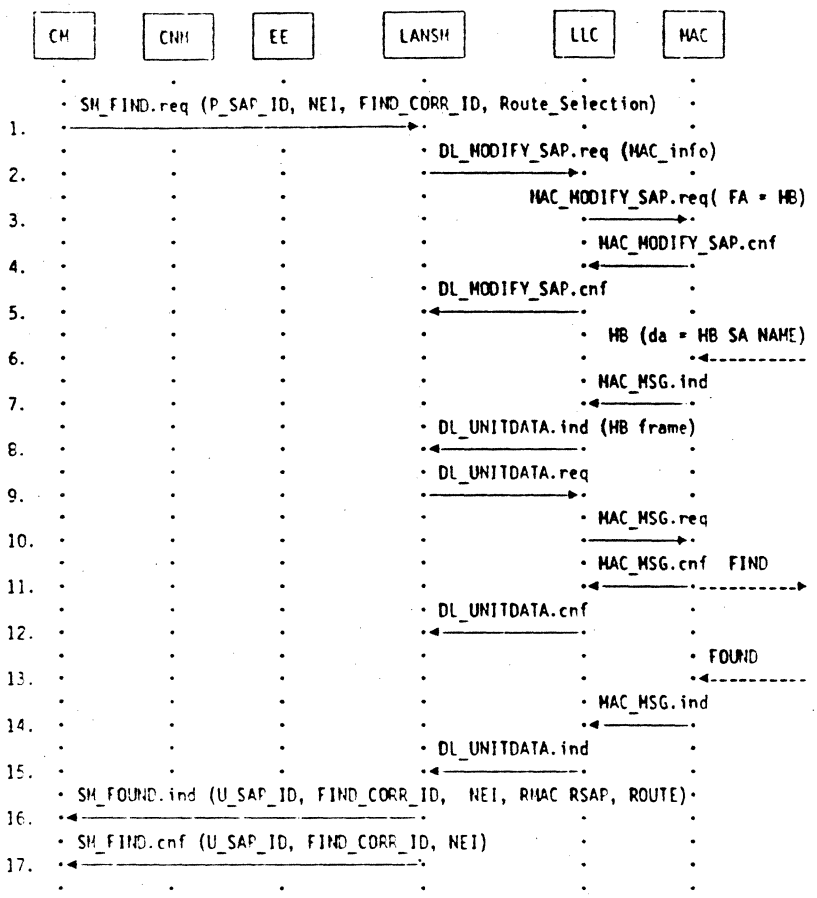


Figure 4-15. Receiving a Heartbeat flow

The following notes are the descriptions of the numbered primitive flows shown above.

1 Connection mgr. requests LANSM to send a FIND frame to locate a particular application identified by Network Entity Identifier (NEI).

The Connection mgr includes a FIND_CORR_ID to be used for routing of the confirm and future SM_FOUND_ind primitives. The FIND_CORR_ID is used to correlate FIND/FOUND frames (e.g. if the FIND includes a Group Name and the FOUND contains a specific name).

The Route_Selection byte specifies a particular algorithm to be used by the LANSM in "best route" selection. This is only applicable if route selection is done in the LANSM.

2-5 The LLC causes the MAC to "open" the heartbeat functional address and to begin monitoring the ring for a heartbeat frame for a period of time (defined by the protocol.)

6-8 In this example, a heartbeat (for the desired application) is received and is passed to the LLC, which in turn passes it to the LANSM (via the Name Mgmt. SAP).

9-12 The LANSM interprets the frame and builds a Find frame using the address provided in the Heartbeat frame, passes it to the LLC (via the Name Mgmt. SAP), which in turn causes the MAC to send the frame to the Network.

13-15 One (or more) Found frame(s) may be received from the remote station and are passed up to the LANSM.

Note: In the event of two or more FOUND frames that contain different Source Address values are received, the LANSM may notify the CNM manager (using an LSA CNM primitive.)

16 The "best route" is selected from the FOUND frame(s) received.

Note: If the "best route" selection is done in the LANSM, the DL_FOUND.ind is sent to the user containing the best route. If the route selection is done by the LANSM user, either a list of routes may be sent in one DL_FOUND.ind to the LANSM user or a series of DL_FOUND.ind (one route per FOUND.ind) may be sent to the LANSM user. A LANSM link configuration parameter (Best_Route_Selection) identifies if the selection is done in the LANSM or not.

The FIND.request specifies the route selection algorithm.

17 The confirm may be used to signify the last FOUND.ind for a given FIND.

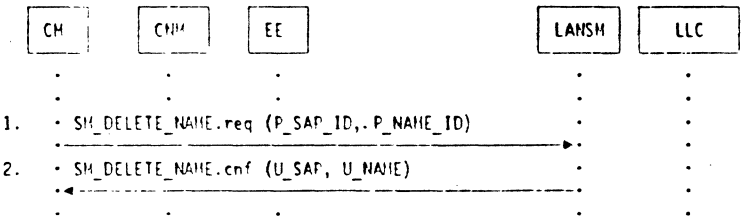


Figure 4-16. Deleting a name flow

1 Connection Manager requests the station manager to delete name identified by the P_NAME_ID.

Note: If the LANSM is heartbeating for the name (to be deleted), the heartbeating is stopped.

Note: If the P_NAME_ID is all zeros, then all the names associated with the SAP are deleted.

2 LANSM sends a confirm, indicating that the name has been deleted.

4.7 LAN Station Manager Primitive Definition

4.7.1 Common Elements of the LSA Primitives

To promote commonality, each primitive will be structured such that common parameters, common fields, etc. are specified in the same way and at the same offsets. The goal is to provide a common base for each primitive that can be customized by adding protocol-specific information.

4.7.2 Primitive types

There are four types of LSA primitives:

Request Confirm
Response Indication

All primitives are divided into two sections; an Interface Control Information (ICI) section and an Interface Data (ID) section.

The Interface Control Information section is a variable length data structure that contains the parameters that tailor the function of the primitive.

The Interface Data section contains the Data associated with this primitive. It contains such information as configuration data or I-field data.

The generic formats of these primitive types are described below.

Request, Indication, and Response format

All three of these primitive types have the same format.

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code
4	2	Interface Data field length
6	1	Selector
7	1	ID Type
8	4	Identifier (ID)
12	var	Primitive and protocol specific information

Interface data:

Byte	Length	Description
0	n	Data, parameters, etc.

Confirm format

The confirm primitive type shares the same base format as the other primitive types but adds information that is pertinent to reporting completion status.

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code
4	2	Interface Data field length
6	1	Selector
7	1	ID Type
8	4	Identifier (ID)
12	12	Common Status
24	var	Primitive and protocol specific information

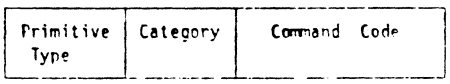
Interface data:

Byte	Length	Description
0	n	Status information, parameters, data, etc.

Common ICI parameter definitions

ICI length Interface Control Information length is the total byte count of the ICI including itself.

Primitive Code This is the hexadecimal encoding identifying a specific primitive.



Bits 0 - 3 Primitive Type

X'0' = Request
X'4' = Indication
X'8' = Response
X'C' = Confirm

Bits 4 - 7 Category

X'C' = Management primitive
X'D' = Normal service primitive
X'E' = CNM primitive

Bits 8 - 15 Command Code

This is a hexadecimal encoding identifying the function to be performed.

Interface Data field length This is the total byte count of the Interface Data area of the primitive. If this field is set to zero, no interface data exists.

Selector

This field identifies the layer to which this primitive is directed. This allows multiple LSA entities to share a common input queue or entry point.

Layer ID Bits 0-3:

X'2' = SM
X'6' = DLC LLC
X'A' = MAC
X'B' = Managed Appl. Entity

Bits 4-7: Reserved

ID Type

This field defines the contents of the Identifier field.

0 = ID field contains zeros and this primitive will be handled by the Entity Manager within the entity.

1 = ID field contains a u_CEP_id, Invoke Identifier, or Name Identifier.

2 = ID field contains a p_CEP_id, Invoke Identifier, or Name Identifier.

3 = ID field contains a u_SAP_id

4 = ID field contains a p_SAP_id

5 = ID field contains zeros and this primitive relates to the entire physical link (not to a particular CEP or SAP).

6 = ID field contains zeros and this primitive relates to all stations and not to a particular CEP or SAP.

7 = ID field contains zeros and this primitive relates to all SAPs.

8 = Reserved

9 = ID field contains a p_SAP_id and this primitive relates to all stations within the identified SAP.

Identifier (ID)

Based on the ID type field, this contains either zeros (Entity Manager within the entity) or a Service Access Point Identifier (SAP_id) alias, or an identifier for a particular resource (Connection End Point (CEP), Invoke instance, or Name instance.)

Common Status

This field contains the completion code (and other information) which indicates whether successful completion has occurred, or whether some error has been encountered. In addition the field contains information identifying the Entity that detected the error, the type of error, and logging information.

Primitive and protocol specific information

Contains information that is unique to either the particular primitive or the particular protocol.

Note: This is not a free-form field. The individual primitives will be explicitly defined. This is simply a generic format concept.

4.8 Primitive formats definitions

4.8.1 SM_ENABLE.request

Function:

Issued by the Entity Enabler to provide the initial configuration to a newly created LANSM instance with link configuration parameters. The primitive carries both the entity instance name and its configuration data.

Parameters:

ICI Generic Header

Entity Instance Name This is the name of this particular entity instance. This is used to identify the instance.

Provider Entity Instance Name This is the name of the invocation of the provider entity that this entity will use. (i.e., the LLC instance name).

User Entity Instance Name This is the name of the invocation of the user entity for which the LANSM will be the provider.

Link Configuration Parameters The configuration parameters for the LAN Station Manager.

Best_Route_Selection This determines if the best route selection process (done on FOUND messages received) is done in the LANSM or not.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0C00')
4	2	Length of data area (= n)
6	1	Selector
7	1	ID Type (= X'00')
8	4	Identifier (= 0)

Interface data:

Byte	Length	Description
0	16	Entity Instance name
16	1	Best route selection = X'00' done by LANSM = X'01' not done by LANSM
17	m	Configuration parameters

4.8.2 SM_ENABLE.confirm

Function:

This primitive is used to reply to a SM_ENABLE.req.

Parameters:

ICI Generic Header

ID Type The ID Type is zero indicating that the confirm is directed to the entity manager issuing the SM_ENABLE.req (Entity Enabler). This primitive is returned to the Entity Enabler using the same underlying transport over which the entity received the SM_ENABLE.req

Entity Instance Name This parameter is returned in the SM_ENABLE.cnf as a correlator to match the confirm with the previous request.

Link Configuration Parameters The configuration parameters are returned in the event of a negative completion code. The format is the same as the link configuration description in SM_ENABLE.request.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CC00')
4	2	Length of data area
6	1	Selector
7	1	ID Type (=X'00')
8	4	Identifier (= 0)
12	12	Common status
24	16	Entity Instance name

Interface data:

Byte	Length	Description
0	n	Configuration data (if CC ≠ 0)

Completion Codes:

Generic Completion Codes

4.8.3 SM_DISABLE.request

Function:

To inform the entity that its services are no longer required by the user.

This primitive is issued by the Entity Enabler to clear the configuration information stored by the entity. After receiving this primitive, the only primitive that the entity will accept is SM_ENABLE.

This primitive is targeted to the Entity Manager within the entity being disabled.

Parameters:

ICI generic header

ID Type — This field contains zero which indicates that the Identifier field contains zeroes and this primitive is targeted to the Entity Manager within the entity being disabled.

Identifier — This field is reserved in this primitive.

Entity Instance Name — This field contains the Entity Instance Name originally given the entity in the SM_ENABLE.req. This parameter is used as a correlator to allow matching the returning confirm with this request.

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0C01')
4	2	Interface Data field length
6	1	Selector
7	1	ID Type (= X'00')
8	4	Identifier (= 0)
12	16	Entity Instance name

4.8.4 SM_DISABLE.confirm

Function:

This primitive is used to reply to a SM_DISABLE.request.

Parameters:

ICI generic header

ID Type — The ID Type is zero indicating that the Identifier field contains zeroes and that the SM_DISABLE.cnf is directed to the Entity Manager within the entity issuing the SM_DISABLE.req (the Entity Enabler's Entity Manager).

Identifier — This field is reserved in this primitive.

Entity Instance Name — This field contains the name of the entity as received in the SM_DISABLE.req which is used by the Entity Enabler as a correlator to match this confirm with the previous request.

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CC01')
4	2	Interface Data field length
6	1	Selector
7	1	ID Type (= X'00')
8	4	Identifier (= 0)
12	12	Common status
24	16	Entity Instance name

4.8.5 SM_ACTIVATE_SAP.request

Function:

This primitive is used to activate a Service Access Point (SAP) and to identify the particular management functions that are available from the (N+1)_Entity at this SAP. The management functions can be provided to the lower layer entity for:

Just This SAP The management service provided through this SAP is only applicable for the CEPs and connectionless data flow through this SAP.

ALL SAPs The management service provided through this SAP is applicable for this SAP and all other SAPs activated to this entity (with the exception of those SAPs that were activated with "Just This SAP" management services.

SAP Group The management service provided though this SAP is applicable for this SAP and all other SAPs that are identified as belonging to this SAP group. SAP grouping are indicated by a common value in the Service Range field (that value > 1.)

As described, there is no restriction on the number of different SAPs that can be active at any one time to different entities, however, **there can only be one SAP with a Service Range of "ALL SAPs" for a given management Service Type per entity.** The "Just this SAP" Service Range takes precedence over the ALL SAPs range for the SAP being activated from a combined Management/Data User entity. A SAP Group identification takes precedence over the ALL SAPs range.

Each SAP is identified by its SAP Name which is contained in the Interface Data field.

This primitive is directed to the entity manager (Identifier = zero) and includes the u_SAP_id alias that the User Entity Manager assigned for reference to this SAP.

The Provider Entity Manager assigns its p_SAP_id alias, the p_sap_id is returned in the DL_ACT_SAP.cnf along with the u_SAP_id alias which is used as a correlator.

Parameters:

ICI Generic Header

U_SAP_ID This is the SAP_ID alias that the user assigns to identify the SAP.

Service Range identifies whether the services identified by the Service Type field are available for all SAPs, or just this one, or for a SAP group.

Service Type Identifies the type of User service functions available through this SAP. Notice that the Service Type can identify a Management service alone, a Data User service alone, or a combined Management/Data User service. The identified service functions are:

- XID Manager
- Connection Manager
- CNM Manager
- Error Log Manager
- Data User

User Path Identifier Addressing information to allow the provider to establish the logical association of a queue_id, task_id or other construct to the User's SAP access for this SAP being activated.

Interface Data:

SAP related information

- Encoding_Rule - The rule for encoding the CMIP data carried in the Interface Data of LSA primitives. Currently the only scheme defined is ASN.1.
- Architecture Version - Identifies the version number of the architecture supported. The initial version is X'0001'.
- This field may contain other SAP specific configuration parameters.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0C0D')
4	2	Length of data area (= n)
6	1	Selector
7	1	ID Type (= X'00')
8	4	Identifier (= X'00000000')
12	4	U_SAP_ID
16	1	Service range X'00' = Services for this SAP only X'01' = Services for all SAPs X'02' to X'FF' = Services for SAP Group
17	1	Service type Bit 0 = 1 — Reserved Bit 1 = 1 — XID Manager Bit 2 = 1 — Reserved Bit 3 = 1 — Connection Manager Bit 4 = 1 — CNM Manager Bit 5 = 1 — Error Log Manager Bit 6 = 1 — Reserved Bit 7 = 1 — Data User
18	16	User Path Identifier

Interface data:

Byte	Length	Description
0	2	Architecture Version (X'0001')
2	1	Encoding_Rule Bit 0 = 1—ASN.1 Bit 1 — Bit 7 —Reserved (=0)
3	n	SAP Related Information

4.8.6 SM_ACTIVATE_SAP.confirm

Function:

This primitive is used to reply to a SM_ACTIVATE_SAP.request.

Parameters:

- ICI Generic Header
- u_SAP_id - is used to correlate this confirm with the original request.
- P_SAP_ID - This is the SAP_ID (which the provider assigns) that the provider has associated with the U_SAP_ID that was included in the request.
- Provider Path Identifier - Addressing information to allow the User to establish the logical association of a queue_id, task_id or other construct to the Provider's SAP access for this SAP being activated
- SAP Related Information - This field is used to identify the objects that are "owned" by a particular LME to the LANSM (For example this field may contain Object Class and Object Instance.)

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (X'CC0D')
4	2	Length of data area (= n)
6	1	Selector
7	1	ID Type (= X'03')
8	4	Identifier (U_SAP_ID)
12	12	Common Status
24	4	P_SAP_ID
28	16	Providedr path identifier

Interface data:

Byte	Length	Description
0	n	SAP related information

4.8.7 SM_DEACTIVATE_SAP.request

Function:

To indicate that the user who issued this primitive wishes to terminate the use of the identified SAP.

This primitive is used in two cases:

1. To deactivate a SAP which has been created by a previous SM_ACTIVATE_SAP.req/cnf.
2. To cancel a SM_ACTIVATE_SAP.req before the confirm is received.

Parameters:

- ICI Generic Header
- ID Type - An ID Type of 4 indicates that the P_SAP_ID is for case 1. For case 2, this field will indicate an ID Type of zero which is addressed to the Entity Manager and the SAP will be identified by its U_SAP_ID.
- U_SAP_ID - The SAP ID of the SAP that is being deactivated.

  For consistency, the u_sap_id identifying the SAP being deactivated is also included in case 1.
- SLID (System Log Identifier) - If the SAP is being closed due to an error detected by the user, this field identifies the error that was logged.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0C0E')
4	2	Length of data area (X'0000')
6	1	Selector
7	1	ID Type = X'00' — Entity manager case 2 = X'04' — p_SAP_ID case 1
8	4	Identifier Reserved case 2 P_SAP_ID case 1
12	4	U_SAP_ID
16	4	System log identifier

4.8.8 SM_DEACTIVATE_SAP.confirm

Function:

This primitive is used to reply to a SM_DEACTIVATE_SAP.request.

Parameters:

- ICI Generic Header

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CC0E')
4	2	Length of data area (= X'0000')
6	1	Selector
7	1	ID Type (= X'03')
8	4	U_SAP_ID
12	12	Common status

Completion Codes:

Generic Completion Codes

4.8.9 SM_ADD_NAME.req

Function:

This primitive is used to pass a Network Entity Identifier of a user to the LANSM (to be added to a NEI database.)

The LANSM may Heartbeat this name.

Parameters:

- ICI Generic Header
- p_SAP_id - The SAP_id identifying the SAP activated by the Connection Manager.
- u_Name_id - The identifier (assigned by the Connection Manager) that refers to the name being added.
- Network Entity identifier - The name of the user (e.g. Application) to be added to the data base of Network Entity Identifiers.
- Group Identifier - The group name of the user (e.g. Application) to be added to the data base of Network Entity Identifiers.
- Protocol ID - Specifies the type of protocol stack (e.g., SNA, OSI, etc.) to be associated with the NEI.
- User_LSAP - The user's LSAP address.
- Total_Group_Identifier - The number of group Identifiers included in this request.
- Length_of_Group_Identifiers - The total length of all group Identifiers in this request.
- LLC Name - Identifies a particular instance of LLC managed by the LANSM.
- SNAPPROTID - SNAP Protocol Identifier as defined in IEEE 802.1A.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0E60')
4	2	Interface Data field length
6	1	Layer selector
7	1	ID Type (= X'04')
8	4	Identifier (p_SAP_ID)
12	4	u_NAME_ID

Interface data:

Byte	Length	Description
0	1	ACTION: = X'01' — Send HB frames = X'00' — Don't send HB frames
1	1	LLC name
2	1	Protocol identifier
3	1	User_LSAP
4	2	Network entity identifier length
6	n	Network entity length
6	4	SNAPPROTID
10 + n	2	Total group identifiers (= b)
12 + n	2	Total length of all group identifiers (= j)
14 + n	2	Length of first group identifier (= j1)
16 + n	j1	Group identifier
16 + n + j1	2	Length of second group identifier (= j2)
18 + n + j1 + j2	j2	Group identifier
⋮	⋮	⋮
j-(jb + 2)	2	Length of b'th group identifier (= jb)
j-jb	jb	Group identifier

Action by the LAN Station Manager:

Upon receipt of this request, the NEI and related information is added to a NEI data base (in the LANSM).

4.8.10 SM_ADD_NAME.cnf

Function:

This primitive is used to reply to a SM_ADD_NAME.req and to pass the P_Name value to the user.

Parameters:

- ICI Generic Header
- u_NAME_id - Identifier (assigned by the Connection Manager) that refers to the name being added.
- p_NAME_id - Identifier (assigned by the LANSM) that refers to a specific name in the Network Entity Identifier data base.
- Interface Data - In the event of invalid interface data field parameter, they are returned.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE60')
4	2	Length of data area
6	1	Layer selector
7	1	ID Type (= X'01')
8	4	u_NAME_ID
12	12	Common status
24	4	p_NAME_ID

Interface data:

Byte	Length	Description
0	n	Parameters

Completion Codes:

Generic Completion Codes

4.8.11 SM_DELETE_NAME.req

Function:

This primitive requests the LANSM to delete a Name (and any related information) from a LANSM Network Entity Identifier data base.

Note: If the P_Name value is all zeros, then all the names associated with the SAP (identified by p_SAP_id) are deleted.

Note: If the LANSM is Heartbeating for the name being deleted, the Heartbeating is stopped.

Parameters:

- ICI Generic Header
- p_NAME_id - Used to refer to a specific name in the Network Entity Identifier data base.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0E61')
4	2	Interface Data field length
6	1	Layer selector
7	1	ID Type (= X'02')
8	4	p_NAME_ID

Action by the LAN Station Manager:

The LANSM performs the necessary actions to remove a name (and any related information) or names from its Network Entity Identifier data base.

4.8.12 SM_DELETE_NAME.cnf

Function:

This primitive is used to reply to a SM_Delete_Name.req.

Parameters:

- ICI Generic Header
- u_NAME_id - the identifier (assigned by the Connection Manager) that refers to the name being deleted.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE61')
4	2	Interface Data field length
6	1	Layer selector
7	1	ID Type (= X'01')
8	4	u_NAME_ID
12	12	Common status

Completion Codes:

- Generic Completion Code

4.8.13 SM_FIND.req

Function:

This primitive requests the LANSM to discover the MAC Address, SAP Address and/or route of a remote station that "owns" a specific Network Entity Identifier.

Parameters:

- ICI Generic Header
- p_SAP_id - The SAP_id identifying the SAP activated by the Connection Manager.
- FIND_CORR_ID - This identifier provides a method to correlate a FIND request with FOUND indications.

  Interface Data
- Route_Selection - Identifies the algorithm to be used in route selection when the FOUND frames are received.

  Note: Some examples of Route Selection algorithms are:

  - First Found - Selects the route in the first FOUND response received.
  - First Found n Routes - Selects the route in the first FOUND response with less than MAX_ROUTE_DESIGNATORS.
  - Least Route Designators - Selects the route in the FOUND frame with the fewest route designators within FOUND_WINDOW seconds.
  - First Found LF - Selects the route in the FOUND frame with a frame size larger than MIN_FRAME_SIZE.
  - LF Within Window - Selects the route in the FOUND frame with the largest frame size within FOUND_WINDOW seconds.
  - First Found LF n Routes - Selects the route with fewer than MAX_ROUTE_DESIGNATORS with the largest frame size greater than MIN_FRAME_SIZE.
  - Least Route LF Within Window - Selects the route in the FOUND frame with the fewest route designators, with a frame size larger than MIN_FRAME_SIZE, and within FOUND_WINDOW seconds.
  - N Routes LF Within Window - Selects the route in the FOUND frame with the less than MAX_ROUTE_DESIGNATORS, with the largest frame size larger, and within FOUND_WINDOW seconds.
- Frame_size - The frame size to be used in the FIND frame.
- Network Entity Identifier - the NEI of a service available at a remote station.
- MAC Address - If the MAC address is known by the user, it may be included and the function of the FIND is to discover only the route.
- MAX_ROUTE_DESIGNATORS - The maximum number of route designators to be used for some Best Route selection algorithms.
- FOUND_WINDOW - The number of seconds some Best Route selection algorithms monitor for FOUND messages.
- MIN_FRAME_SIZE - The minimum frame size, used in some Best Route selection algorithms.

- Group Network Entity Identifier - The group name of the user (e.g. Application) to be added to the data base of Network Entity Identifiers.
- Protocol ID - Specifies the type of protocol stack (e.g., SNA, OSI, etc.) to be associated with the NEI.
- HBWAIT - Specifies that the LANSM is to wait for a Heatbeat frame to be received for a NEI before sending a FIND or use normal procedure (wait for a Heartbeat frame for a period of time and then send a FIND.)

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0E62')
4	2	Length of data area
6	1	Selector
7	1	ID Type (= X'04')
8	4	p_SAP_ID
12	4	FIND_CORR_ID

Interface data:

Byte	Length	Description
0	1	Route_Selection = X'01' — First found = X'02' — First found N routes = X'03' — Least route designators = X'04' — First found LF = X'05' — LF within window = X'06' — First found LF N routes = X'07' — Least route LF within window = X'08' — N route LF within window
1	2	Frame size
3	1	HBWAIT X'01' = — Wait for a Heart Beat X'02' = — Normal
4	6	MAC address
10	1	Protocol ID
11	2	MAX_ROUTE_DESIGNATORS
13	2	FOUND_WINDOW
15	2	MIN_FRAME_SIZE
17	2	Network entity identifier length
19	n	Network entity identifier
19 + n	2	Goup network entity indentifier length
21 + n	m	Group network entity identifier

Action by the LAN Station Manager:

Upon receipt of this primitive, the LANSM monitors the network for a period of time (HB_Delay). If a HB is received, a Discovery frame is built and sent using the MAC address received in the HB. If no HB is received, the Discovery frame is built and sent with the locate functional address.

If the MAC Address is included in the ID field, a FIND frame is sent (to discover the route) without monitoring for a HB frame.

4.8.14 SM_FIND.cnf

Function:

This primitive is used to reply to the SM_FIND.req.

Parameters:

- ICI Generic Header
- FIND_CORR_ID - This identifier is used to correlate this confirm to a SM_FIND.request.
- Interface Data - In the event of invalid interface data field parameter (defined in the request), they are returned.
- Reply_Code - Indicates if No Found messages were received.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE62')
4	2	Length of data area
6	1	Selector
7	1	ID Type (= X'03')
8	4	U_SAP_ID
12	12	Common status
24	4	FIND_CORR_ID

Interface data:

Byte	Length	Description
0	1	Reply code X'00' = No Found messages received X'01' Found messages received
1	n	Parameters

Completion Codes:

Generic Completion Code

4.8.15 SM_FOUND.ind

Function:

This primitive is used to pass discovered MAC, SAP, and routing information (received on a FOUND frames) to the LANSM User.

Note: If "Best route" selections is done by the LANSM, then only the "Best route" (MAC, SAP, Route) would be passed in the interface data. If "Best route" selection is done by the LANSM user, then all the routes (MAC, SAP, Route) received are passed in the interface data.

Parameters:

- ICI Generic Header
- u_SAP_id - is used to correlate this confirm with the original request.
- Number of Routes - the number of FOUND messages received (If best route selection is done in the LANSM, this value will be 1).
- FIND_CORR_ID The identifier is used to correlate a FOUND.indication to a FIND.request.

Interface Data:

If "Best_Route" selection is done by the LANSM user, then information from one or more FOUND frames may be sent in one indication. If information from more than one FOUND frame is passed up (in one FOUND.ind) then the remote MAC address, remote SAP address, length of route, and routing information will be repeated in the interface data field for each FOUND frame received.

- Remote MAC - The remote MAC address of the station who issued a FOUND frame.
- Remote SAP address - The remote SAP address of the station who issued a FOUND frame.
- Length of Route - The length of the routing information on a received FOUND frame.
- Routing information - The routing information received in a FOUND frame.
- NEI - The Network Entity Identifier received in the FOUND frame.
- SNAPPROTID - SNAP Protocol Identifier defined om IEEE 802.1A.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'4E63')
4	2	Length of data area (10 + n + m)
6	1	Selector
7	1	ID Type (= X'03')
8	4	u_SAP_ID
12	1	Number of routes)MAC, SAP, Route)
13	4	FIND_CORR_ID

Interface data:

Byte	Length	Description
0	6	Remote MAC address
6	1	Remote SAP address
7	1	Length of route
8	4	SNAPPROTID
12	n	Routing information
12 + n	2	Network entity identifier length
14 + n	m	Network entity identifier

4.8.16 SM_INVOKE.req

Function:

This primitive is used by the CNM manager to pass CMIP data to the LANSM and causes the LANSM to send an RO-INVOKE.

Parameters:

- ICI Generic Header
- p_SAP_id - The SAP_id identifying the SAP activated by the CNM Manager.
- u_INV_id - The identifier (assigned by the CNM Manager) that refers to the Invoke instance.
- LLC Name - Identifies a particular local instance of LLC (managed by the LANSM.)
- MAC Address - The MAC address of the remote node (destination of the INVOKE.)
- SAP Address - The SAP address of the remote node (destination of the INVOKE.)
- Route - Routing information for the remote node (destination of the INVOKE.)
- CMIP Data - This field contains an Operation identifier (which identifies the CMIP operation) and objects with their attributes.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0E70')
4	2	Interface Data field length
6	1	Layer selector
7	1	ID Type (= X'04')
8	4	p_SAP_id
12	4	u_INV_id

Interface data:

Byte	Length	Description
0	1	LLC name
1	1	SAP address
2	6	MAC address
8	1	Reserved (= X'00')
9	1	Length of route
10	m	Route
10+m	n	CMIP data

Action by the LAN Station Manager:

Upon receipt of this request, an INVOKE_ID is assigned, a RO INVOKE frame is built and sent to the LLC. The LANSM also assigns a P_INV_ID that is associated with U_INV_ID passed in this request.

4.8.17 SM_INVOKE.cnf

Function:

This primitive is used to reply to a SM_INVOKE.req and to pass the P_INV_ID value to the user.

Parameters:

- ICI Generic Header
- u_INV_id - Identifier (assigned by the CNM Manager) that refers to the INVOKE instance.
- p_INV_id - Identifier (assigned by the LANSM) that refers to a specific INVOKE instance.
- Interface Data - In the event of invalid interface data field parameter, the parameters are returned.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE70')
4	2	Length of data area
6	1	Layer selector
7	1	ID Type (= X'01')
8	4	u_INV_id
12	12	Common status
24	4	p_INV_id

Interface data:

Byte	Length	Description
0	n	Parameters

Completion Codes:

Generic Completion Codes

4.8.18 SM_INVOKE.ind

Function:

This primitive indicates to the CNM manager that an RO-INVOKE has been received and is used to pass CMIP data to the CNM Manager.

The primitive is applicable in two cases:

1. Solicited RO-INVOKE - This case pertains to linked replies, where an RO-INVOKE is received and passed to the LANSM as part of a series of linked replies. In this case the primitive is targeted to the U_INV_ID (which was passed in the original SM_INVOKE.request) and carries the P_INV_ID which may be needed in the case of an error or reject condition that would require a reply.
2. Unsolicited RO-INVOKE - This case pertains to the receipt of an RO-INVOKE-EVENT-REPORT. In this case, the primitive is targeted to the U_SAP_ID (CNM SAP) and carries the P_INV_ID (needed in the case of a confirmed event.)

Parameters:

- ICI Generic Header
- IDENTIFIERs
  - U_INV_ID - The identifier (assigned by the LANSM user) that allows direct indexing of the INVOKE indication. CASE 1.
  - U_SAP_ID - The identifier (assigned by the LANSM user) that identifies the LSA SAP used to route this primitive to the LANSM user. CASE 2.
- P_INV_ID - The identifier (assigned by the LANSM) that allows direct indexing of a reply (e.g., INVOKE.response, ERROR.request, etc.)
- LLC Name - This name identifies a particular instance of LLC managed by the LANSM.
- MAC Address - The MAC address of the remote node (destination of the INVOKE.)
- SAP Address - The SAP address of the remote node (destination of the INVOKE.)
- Route - Routing information for the remote node (destination of the INVOKE.)
- CMIP Data - This field contains an Operation identifier (which identifies the CMIP operation) and objects with their attributes.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'4E70')
4	2	Length of data area
6	1	Selector
7	1	ID Type = X'01' — U_INV_ID case 1 = X'03' U_SAP_ID case 2
8	4	Identifier U_INV_ID case 1 U_SAP_Id case 2
12	4	P_INV_ID

Interface data:

Byte	Length	Description
0	1	LLC name
1	1	SAP address
2	6	MAC address
8	1	Reserved (= X'00')
9	1	Length of route
10	m	Route
10 + m	n	CMIP data

4.8.19 SM_INVOKE.rsp

Function:

This primitive is used to respond to an INVOKE.indication, and to pass u_INV_ID to the LANSM.

In the case of an unconfirmed command this primitive may be used to remove and invoke instance after the INVOKE.indication.

Parameters:

- ICI Generic Header
- P_INV_ID - The identifier that allows direct indexing of the INVOKE indication.
- u_INV_ID - The identifier (assigned by the CNM manager) that refers to an INVOKE instance.
  1. In the case of an unconfirmed CMIP command, this field will be reserved and this primitive is used to instruct the LANSM to remove the Invoke Instance identified by the P_INV_ID.
  2. In the case of a confirmed CMIP command, this field contains the U_INV_ID (assigned by the LANSM user) which is being passed to the LANSM.
- ACTION - Instructs the LANSM to keep the invoke instance (in the case of a confirmed command) or to remove it (in the case of an unconfirmed command.)

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'8E70')
4	2	Length of data area
6	1	Selector
7	1	ID Type (= X'02')
8	4	p_INV_id
12	4	u_INV_Id or Reserved

Interface data:

Byte	Length	Description
0	1	ACTION X'00' = Remove invoke instance X'01' = Keep invoke instance

4.8.20 SM_RESULT.req

Function:

This primitive is used by the CNM manager to pass CMIP data to the LANSM and causes the LANSM to send an RO-RESULT.

Parameters:

- ICI Generic Header
- P_INV_id - The identifier (assigned by the LANSM) that refers to the Invoke instance.
- CMIP Data - This field contains an Operation identifier (which identifies the CMIP operation) and objects with their attributes.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0E71')
4	2	Interface Data field length
6	1	Layer selector
7	1	ID type (= X'02')
8	4	p_INV_id

Interface data:

Byte	Length	Description
0	n	CMIP data

Action by the LAN Station Manager:

Upon receipt of this request, a RO RESULT frame is build (using the INVOKE ID in the received frame and sent to the LLC.)

4.8.21 SM_RESULT.cnf

Function:

This primitive is used to reply to a SM_INVOKE.req and to pass the P_INV_ID value to the user.

Parameters:

- ICI Generic Header
- u_INV_id - Identifier (assigned by the CNM Manager) that refers to the INVOKE instance.
- Interface Data - In the event of invalid interface data field parameter, they are returned.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE71')
4	2	Length of data area
6	1	Layer selector
7	1	ID Type (= X'01')
8	4	u_INV_id
12	12	Common status

Interface data:

Byte	Length	Description
0	n	Parameters

Completion Codes:

Generic Completion Codes

4.8.22 SM_RESULT.ind

Function:

This primitive is used to indicate to the CNM manager that an RO-RESULT has been received by the LANSM and to pass CMIP data to the CNM manager.

Parameters:

- ICI Generic Header
- u_INV_ID - The identifier (assigned by the user) that allows direct indexing of the INVOKE indication.
- CMIP Data - This field contains an Operation identifier (which identifies the CMIP operation) and objects with their attributes.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'4E71')
4	2	Length of data area
6	1	Selector
7	1	ID Type (= X'01')
8	4	u_INV_id

Interface data:

Byte	Length	Description
0	n	CMIP data

4.8.23 SM_REJECT.req

Function:

This primitive is used by the CNM manager to cause the LANSM to send an RO-REJECT.

Parameters:

- ICI Generic Header
- P_INV_ID - The identifier that refers to the Invoke instance.
- CMIP Data - This field contains an Operation identifier (which identifies the CMIP operation) and objects with their attributes.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0E72')
4	2	Interface Data field length
6	1	Layer selector
7	1	ID Type (= X'02')
8	4	p_INV_id

Interface data:

Byte	Length	Description
0	n	CMIP data

Action by the LAN Station Manager:

Upon receipt of this request, a RO REJECT frame is build (using the INVOKE ID in the received frame and sent to the LLC.

4.8.24 SM_REJECT.cnf

Function:

This primitive is used to reply to a SM_REJECT.req.

Parameters:

- ICI Generic Header
- u_INV_id - Identifier (assigned by the CNM Manager) that refers to the INVOKE instance.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE72')
4	2	Length of data area
6	1	Layer selector
7	1	ID Type (= X'01')
8	4	u_INV_id
12	12	Common status

Completion Codes:

Generic Completion Codes

4.8.25 SM_REJECT.ind

Function:

This primitive is used to indicate to the CNM Manager that the LANSM has has received an RO-REJECT in response to an RO-INVOKE request.

Parameters:

- ICI Generic Header
- u_INV_ID - The identifier (assigned by the user) that allows direct indexing of the INVOKE indication.
- CMIP Data - This field contains an Operation identifier (which identifies the CMIP operation) and objects with their attributes.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (X'4E72')
4	2	Length of data area
6	1	Selector
7	1	ID Type (= X'01')
8	4	u_INV_id

Interface data:

Byte	Length	Description
0	n	CMIP data

4.8.26 SM_ERROR.req

Function:

This primitive is used by the CNM manager to cause the LANSM to send an RO-ERROR.

Parameters:

- ICI Generic Header
- P_INV_ID - The identifier that refers to the Invoke instance.
- CMIP Data - This field contains an Operation identifier (which identifies the CMIP operation) and objects with their attributes.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0E73')
4	2	Interface Data field length
6	1	Layer selector
7	1	ID Type (= X'02')
8	4	p_INV_id

Interface data:

Byte	Length	Description
0	n	CMIP data

4.8.27 SM_ERROR.cnf

Function:

This primitive is used to reply to a SM_ERROR.req.

Parameters:

- ICI Generic Header
- u_INV_id - Identifier (assigned by the CNM Manager) that refers to the INVOKE instance.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE73')
4	2	Length of data area
6	1	Layer selector
7	1	ID Type (= X'01')
8	4	u_INV_id
12	12	Common status

Completion Codes:

Generic Completion Codes

4.8.28 SM_ERROR.ind

Function:

This primitive is used to indicate to the CNM Manager that the LANSM has received an RO-ERROR in response to an RO-INVOKE request.

Parameters:

- ICI Generic Header
- u_INV_ID - The identifier (assigned by the user) that allows direct indexing of the INVOKE indication.
- CMIP Data - This field contains an Operation identifier (which identifies the CMIP operation) and objects with their attributes.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'4E73')
4	2	Length of data area
6	1	Selector
7	1	ID Type (= X'01')
8	4	u_INV_id

Interface data:

Byte	Length	Description
0	n	CMIP data

4.8.29 SM_GET.req

Function:

This primitive is used to retrieve attribute values of a managed object, from a provider entity.

Parameters:

- ICI Generic Header
- p_SAP_id - The SAP_id identifying the SAP activated by the Communications Network Manager.
- Correlator_id - Used to correlate a request with a confirm. This may be used to handle scoping across the interface.
- CNM Data - This field contains the managed object and associated attribute list.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0E00')
4	2	Length of data area (= n)
6	1	Selector
7	1	ID Type (= X'04')
8	4	p_SAP_id
12	4	Correlator_ID

Interface data:

Byte	Length	Description
0	n	CNM data

4.8.30 SM_GET.cnf

Function:

This primitive is used to reply to the SM_GET.req and pass attribute values to the user.

Parameters:

- ICI Generic Header
- u_SAP_id - The SAP_id identifying the SAP activated by the Communications Network Manager.
- Correlator_id - Used to correlate a request with a confirm. This may be used to handle scoping across the interface.
- Linked_Reply_Indicator - This field indicates to the user three states relating to linked replies.
  1. Not a Linked Reply - Indicating that the CNM data is not part of a Linked Reply.
  2. Linked Reply - Indicating that the CNM data is part of a linked reply.
  3. Last Linked Reply - Indicating that the CNM data is the last of a series of linked replies.
- CNM Data - This field contains the managed object, associated attribute list.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE00')
4	2	Length of data area
6	1	Selector
7	1	ID Type (= X'03')
8	4	U_SAP_ID
12	12	Common status
24	4	Correlator_ID

Interface data:

Byte	Length	Description
0	1	Linked reply indicator X'00' = Not a linked reply X'01' = Linked reply X'02' = Last linked reply
1	1	Reserved (= X'00')
2	n	CNM data

Completion Codes:

Generic Completion Code

4.8.31 SM_SET.req

Function:

This primitive is used to set attribute values of a managed object, in a provider entity.

Parameters:

- ICI Generic Header
- p_SAP_id - The SAP_id identifying the SAP activated by the Communications Network Manager.
- Correlator_id - Used to correlate a request with a confirm. This may be used to handle scoping across the interface.
- CNM Data - This field contains the managed object and associated attribute list.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (X'0E10')
4	2	Length of data area (= n)
6	1	Selector
7	1	ID Type (= X'04')
8	4	p_SAP_ID
12	12	Correlator_ID

Interface data:

Byte	Length	Description
0	n	CNM data

4.8.32 SM_SET.cnf

Function:

This primitive is used to reply to the SM_SET.req. In the case of an error the attributes are returned in the interface data.

Parameters:

- ICI Generic Header
- u_SAP_id - The SAP_id identifying the SAP activated by the Communications Network Manager.
- Correlator_id - Used to correlate a request with a confirm. This may be used to handle scoping across the interface.
- Linked_Reply_Indicator - This field indicates to the user three states relating to linked replies.
  1. Not a Linked Reply - Indicating that the CNM data is not part of a Linked Reply.
  2. Linked Reply - Indicating that the CNM data is part of a linked reply.
  3. Last Linked Reply - Indicating that the CNM data is the last of a series of linked replies.
- CNM Data - In the event of an error, this field will contain the managed object and associated attribute list.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE10')
4	2	Length of data area
6	1	Selector
7	1	ID Type (= X'03')
8	4	U_SAP_ID
12	12	Common status
24	4	Correlator_ID

Interface data:

Byte	Length	Description
0	1	Linked reply indicator X'00' = Not a linked reply X'01' = Linked reply X'02' = Last linked reply
1	1	Reserved (X'00')
2	n	CNM data

4.8.33 SM_EVENT.ind

Function:

This primitive is used to notify a user that an Event pertaining to a managed object has occurred.

Parameters:

- ICI Generic Header
- u_SAP_id - The SAP_id identifying the SAP activated by the Communications Network Manager.
- CNM Data - This field contains the managed object and associated attribute list.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'4E20')
4	2	Length of data area (= n)
6	1	Selector
7	1	ID Type (= X'03')
8	4	u_SAP_id

Interface data:

Byte	Length	Description
0	n	CMIP data

4.8.34 SM_ACTION.req

Function:

This primitive is used to request the provider perform an action on a set of managed object.

Parameters:

- ICI Generic Header
- p_SAP_id - The SAP_id identifying the SAP activated by the Communications Network Manager.
- Correlator_id - Used to correlate a request with a confirm. This may be used to handle scoping across the interface.
- CNM Data - This field contains the managed object and associated attribute list.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'E30')
4	2	Length of data area (= n)
6	1	Selector
7	1	ID Type (= X'04')
8	4	p_SAP_ID
12	4	Correlator_ID

Interface data:

Byte	Length	Description
0	n	CNM data

4.8.35 SM_ACTION.cnf

Function:

This primitive is used to reply to the SM_ACTION.req.

Parameters:

- ICI Generic Header
- u_SAP_id - The SAP_id identifying the SAP activated by the Communications Network Manager.
- Correlator_ID - Used to correlate a request with a confirm. This may be used to handle scoping across the interface.
- Linked_Reply_Indicator - This field indicates to the user three states relating to linked replies.
  1. Not a Linked Reply - Indicating that the CNM data is not part of a Linked Reply.
  2. Linked Reply - Indicating that the CNM data is part of a linked reply.
  3. Last Linked Reply - Indicating that the CNM data is the last of a series of linked replies.
- CNM Data - In the event of an error, this field contains the managed object and associated attribute list.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE30')
4	2	Length of data area
6	1	Selector
7	1	ID Type (= X'03')
8	4	U_SAP_ID
12	12	Common status
24	4	Correlator_ID

Interface data:

Byte	Length	Description
0	1	Linked reply indicator X'00' = Not a linked reply X'01' = Linked reply X'02' = Last linked reply
1	1	Reserved (= X'00')
2	n	CNM data

Completion Codes:

Generic Completion Code

4.8.36 SM_CREATE.req

Function:

This primitive is used to request the provider to create a new managed object instance.

Parameters:

- ICI Generic Header
- p_SAP_id - The SAP_id identifying the SAP activated by the Communications Network Manager.
- Correlator_ID - Used to correlate a request with a confirm.
- CNM Data - This field contains the managed object and associated attribute list.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0E40')
4	2	Length of data area (= n)
6	1	Selector
7	1	ID Type (= X'04')
8	4	p_SAP_ID
12	4	Correlator_ID

Interface data:

Byte	Length	Description
0	n	CMIP data

4.8.37 SM_CREATE.cnf

Function:

This primitive is used to reply to the SM_CREATE.req.

Parameters:

- ICI Generic Header
- u_SAP_id - The SAP_id identifying the SAP activated by the Communications Network Manager.
- Correlator_ID - Used to correlate a request with a confirm.
- CNM Data - This field contains the managed object, associated attribute list, and attribute values if the create was not successful.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE40')
4	2	Length of data area
6	1	Selector
7	1	ID Type (= X'03')
8	4	U_SAP_ID
12	12	Common status
24	4	Correlator_ID

Interface data:

Byte	Length	Description
0	n	CNM data

Completion Codes:

Generic Completion Code

4.8.38 SM_DELETE.req

Function:

This primitive is used to request the provider to delete a managed object instance.

Parameters:

- ICI Generic Header
- p_SAP_id - The SAP_id identifying the SAP activated by the Communications Network Manager.
- Correlator_ID - Used to correlate a request with a confirm. This may be used to handle scoping across the interface.
- CNM Data - This field contains the managed object and associated attribute list.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'0E50')
4	2	Length of data area (= n)
6	1	Selector
7	1	ID Type (= X'04')
8	4	p_SP_id
12	4	Correlator_ID

Interface data:

Byte	Length	Description
0	n	CNM data

4.8.39 SM_DELETE.cnf

Function:

This primitive is used to reply to the SM_DELETE.req.

Parameters:

- ICI Generic Header
- u_SAP_id - The SAP_id identifying the SAP activated by the Communications Network Manager.
- Correlator_ID - Used to correlate a request with a confirm. This may be used to handle scoping across the interface.
- Linked_Reply_Indicator - This field indicates to the user three states relating to linked replies.
  1. Not a Linked Reply - Indicating that the CNM data is not part of a Linked Reply.
  2. Linked Reply - Indicating that the CNM data is part of a linked reply.
  3. Last Linked Reply - Indicating that the CNM data is the last of a series of linked replies.
- CNM Data - This field contains the managed object, associated attribute list, and attribute values if the delete was not successful.

IDU FORMAT

Interface Control Information:

Byte	Length	Description
0	2	ICI length (including this field)
2	2	Primitive Code (= X'CE50')
4	2	Length of data area
6	1	Selector
7	1	ID Type (= X'03')
8	4	U_SAP_ID
12	12	Common status
24	4	Correlator_ID

Interface data:

Byte	Length	Description
0	1	Linked reply indicator X'00' = Not a linked reply X'01' = Linked reply X'02' = Last linked reply
1	1	Reserved (= X'00')
2	n	CNM data

Completion Codes:

Generic Completion Code

4.9 Error Handling Overview

The LSA Error Handling mechanism defines the error/status reporting mechanism between two adjacent layer entities in a communications structure.

Its purpose is for the service provider (the lower layer entity) to inform the service user (the upper layer entity) the status of the user's request or any abnormal conditions detected by the service provider. The Completion Status and Error Return Codes are used on several occasions:

1. If an error or exceptional condition is detected during the execution of a request primitive, then the error code is carried as the Completion Codes in the corresponding confirm primitive.
2. If an error or exception condition is asynchronously detected by the service provider while no related request from the service user is outstanding, then the error code is reported as a part of the common status in a provider-initiated indicate primitive.

Note: In LSA it is assumed that the service user does not know nor care how many layers of service there are below it. Therefore the completion (error/return) codes are mapped by each successive layer into a layer code. If the reported error has been logged by the lower layer(s) previously, then the System Log Identifier will be passed to the user as well so that the user can find out more information about this error from the system's error log if needed.

4.10 LSA Common Status Structures

This section describes the Common Status structures for the LSA (Lower-Layer Service Architecture) as shown in Figure 1. Its field indicates whether an error occurred or not. It contains information identifying the Entity that detected the error, the type of error, and logging information.

(Error/return) codes are carried either as the Completion Code field of a confirm primitive, or are included as data on one the indication primitives.

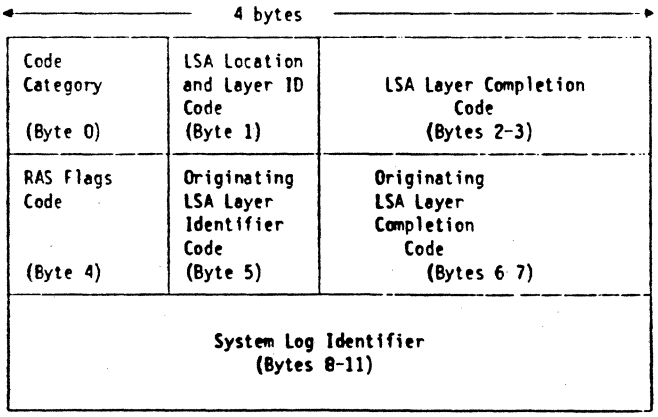


Figure 4-17. The LSA Common Status Structures

The following list the format for Common Status return codes.

Byte 0: Completion Code Category

X'00.' - **LSA Request Success:** This category indicates that the request execution has completed successfully no further information or action required.

X'08' - **LSA Request Reject:** This category indicates that the request was delivered to the intended component and was understood and supported, but not executed.

X'10' - **LSA Request Error:** This category indicates that the request was delivered to the intended component, but could not be interpreted or processed. This condition represents a mismatch of protocol capabilities

X'20' - **State Error:** This category indicates a primitive sequence error that is not allowed for the receiver's current control state.

X'40' - **Usage Error:** This category indicates that the value of a field or combinations of fields in the request violates architectural rules or previously selected options. These errors are independent of the current state of the session. This may result from the failure of the sender to enforce session rules. Detection by the receiver of each of these errors is optional.

X'80' - **Permanent Error:** This category indicates that the request could not be delivered to the intended receiver, because of path outage, an invalid sequence of activation request, or a protocol data unit error. The provider's execution cannot continue, and will only accept RAS or close-down types of commands from the user.

Byte 1: LSA Layer Location and Layer Identifier (ID) Code

This field byte code is split into two parts, which together make up Locations (Local, Path or Line, Remote) and (N)_Layers (ID) identifier code.

Bits 0-3: Layer Location of error

X'1' - Local provider related error or status.

X'2' - Path or line related error or status.

X'3' - Remote provider related error or status..

Bits 4-7: LSA (N)_Layer Identifier (ID) Code

X'6' - LAN LLC

Bytes 2-3: LSA Layer Completion Code

The (N)-layer passes a LSA error/return code to the (N+1)-layer. The (N+1) layer changes this field to N+1 error code but leaves the originating error code alone. This allows each layer to only understand the completion statuses of the next lower layer.

Byte 4: Reserved for future use

Byte 5: Originating LSA Layer Identifier (ID) Code

Same as bits 4 thru 7 of byte 1 field of the LSA Common Status Structures.

Bytes 6-7: Originating LSA Layer Completion Code

The (N)-layer detects an error by itself. The completion code is captured here and will not be modified by the (N+1) layer. (See description of bytes 2-3 of the LSA Common Status Structure.)

Bytes 8-11: System Log Identifier

The System Log Identifier is a system unique identifier that relates this primitive to information that has been sent to the RAS/CNM Manager. This field is only presented byte 4 (RAS Flag Code) status attached. It contains the identifier that was obtained by the layer which initiated the original LSA status cause code.

4.11 LAN Station Manger Completion Codes

The completion codes for the LAN Station Manager will be identified by the block of X'8x' for the LSA Layer completion codes.

Chapter 5. Error Code Description

5.1 Overview

The LSA Error Handling mechanism defines the error/status reporting mechanism between two adjacent layer entities in a communications structure.

Its purpose is for the service provider (the lower layer entity) to inform the service user (the upper layer entity) the status of the user's request or any abnormal conditions detected by the service provider. The Completion Status and Error Return Codes are used on several occasions:

1. If an error or exceptional condition is detected during the execution of a request primitive, then the error code is carried as the Completion Codes in the corresponding confirm primitive.
2. If an error or exception condition is asynchronously detected by the service provider while no related request from the service user is outstanding, then the error code is reported as a part of the common status in a provider-initiated indicate primitive.

Note: In LSA it is assumed that the service user does not know or care how many layers of service there are below it. Therefore the completion (error/return) codes are mapped by each successive layer into a layer code. If the reported error has been logged by the lower layer(s) previously, then the System Log Identifier will be passed to the user as well so that the user can find out more information about this error from the system's error log if needed.

5.2 LSA Common Status Structures

This section describes the Common Status structures for the LSA (Lower-Layer Service Architecture) as shown in Figure 5-1. Its field indicates whether an error occurred or not. It contains information identifying the Entity that detected the error, the type of error, and logging information.

(Error/return) codes are carried either as the Completion Code field of a confirm primitive, or are included as data on one the indication primitives.

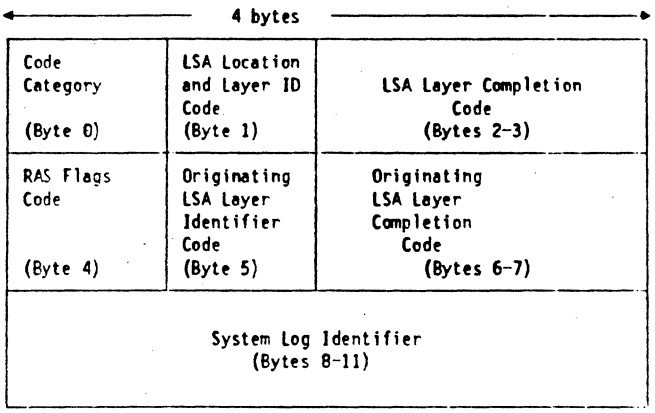


Figure 5-1. The LSA Common Status Structures

The following list the format for Common Status return codes.

Byte 0: Completion Code Category

X'00' - LSA Request Success: This category indicates that the request execution has completed successfully no further information or action required.

X'08' - LSA Request Reject: This category indicates that the request was delivered to the intended component and was understood and supported, but not executed.

X'10' - LSA Request Error: This category indicates that the request was delivered to the intended component, but could not be interpreted or processed. This condition represents a mismatch of protocol capabilities

X'20' - State Error: This category indicates a primitive sequence error that is not allowed for the receiver's current control state.

X'40' - Usage Error: This category indicates that the value of a field or combinations of fields in the request violates architectural rules or previously selected options. These errors are independent of the current state of the session. This may result from the failure of the sender to enforce session rules. Detection by the receiver of each of these errors is optional.

X'80' - Permanent Error: This category indicates that the request could not be delivered to the intended receiver, because of path outage, an invalid sequence of activation request, or a protocol data unit error. The provider's execution cannot continue, and will only accept RAS or close-down types of commands from the user.

Byte 1: LSA Layer Location and Layer Identifier (ID) Code

This field byte code is split into two parts, which together make up Locations (Local, Path or Line, Remote) and (N)_Layers (ID) identifier code.

Bits 0-3: Layer Location of error

X'1' - Local provider related error or status.

X'2' - Path or line related error or status.

X'3' - Remote provider related error or status..

Bits 4-7: LSA (N)_Layer Identifier (ID) Code

X'6' - LAN LLC

Bytes 2-3: LSA Layer Completion Code

The (N)-layer passes a LSA error/return code to the (N+1)-layer. The (N+1) layer changes this field to N+1 error code but leaves the originating error code alone. This allows each layer to only understand the completion statuses of the next lower layer.

Byte 4: Reserved for future use

Byte 5: Originating LSA Layer Identifier (ID) Code

Same as bits 4 thru 7 of byte 1 field of the LSA Common Status Structures.

Bytes 6-7: Originating LSA Layer Completion Code

The (N)-layer detects an error by itself. The completion code is captured here and will not be modified by the (N+1) layer. (See description of bytes 2-3 of the LSA Common Status Structure.)

Bytes 8-11: System Log Identifier

The System Log Identifier is a system unique identifier that relates this primitive to information that has been sent to the RAS/CNM Manager. This field is only presented byte 4 (RAS Flag Code) status attached. It contains the identifier that was obtained by the layer which initiated the original LSA status cause code.

5.3 LAN Station Manger Completion Codes

The completion codes for the LAN Station Manager will be identified by the block of X'8x' for the LSA Layer completion codes.

Chapter 6. Dynamic Address Resolution and Route Discovery

Note

This chapter is subject to change to align it with the route discovery function being developed in project IEEE 802.5 working group.

Dynamic Address Resolution and Route Discovery (or, more simply, Discovery) is a protocol which provides LAN stations with both a directory service and a routing service. That is, by requesting DLC.LAN.MGR to perform Discovery, a LAN station can, given the network entity identifier of a remote network entity, determine not only a MAC address and LSAP through which the entity is accessible, but also, if either the station or the entity lie on a source routing bridged LAN, a route. Discovery uses a combination of HeartbeatRequest, Heartbeat, Find, and Found LLC frames.

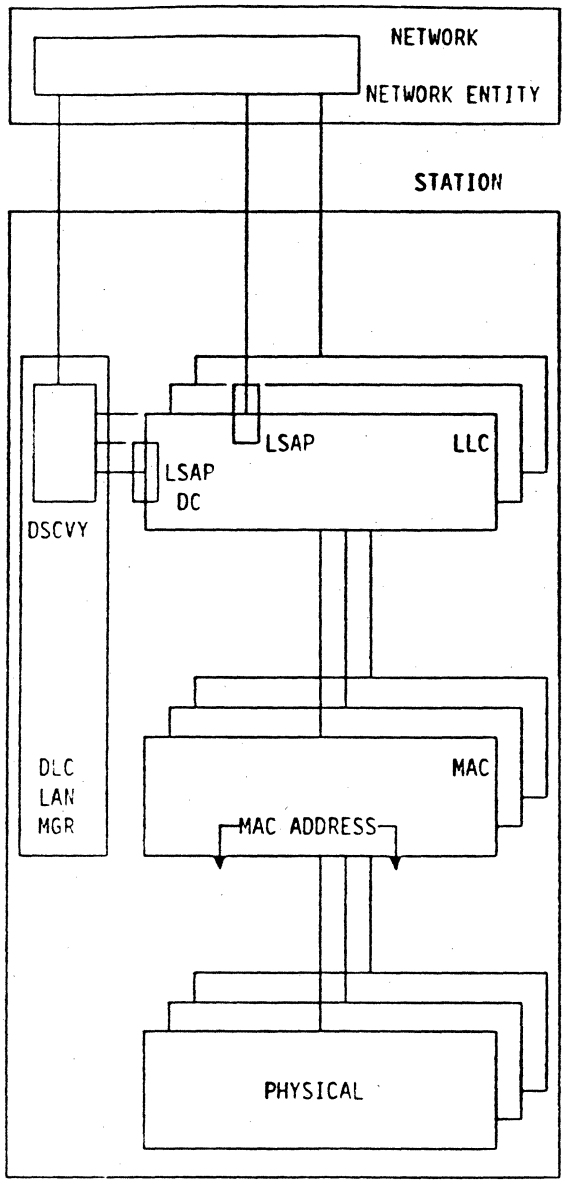


Figure 6-1. Layering

6.1.1 Terms

See Figure 6-1 for an illustration of the definitions of some terms associated with layering. See *IBM Token Ring Architecture Reference* for a discussion of the functions of the DLC.LAN.MGR.

Station — **Consists of one DLC.LAN.MGR and the (possibly multiple) instances of the LLC and MAC sublayers and the Physical layers it manages. All these Physical layer instances must be attached to the same bridged LAN under non-failure conditions.**

Network Entity	Defined in the *OSI Basic Reference Model*, ISO 7498-1984(E) as an active element within the network layer of the OSI layering. The Discovery architecture extends this definition of network entity to include an active element in any layer which sits on the DLC layer. A network entity can be thought of as any entity directly using the DLC. One network entity may communicate through multiple stations.
Network Entity Identifier	A permanent identifier for an entity. Thus, a network entity identifier identifies a subset of functions which interface with the DLC layer.
Regular Identifier	A network entity identifier intended to refer to a single instance of the entity. Each regular identifier refers to one and only one instance of a network entity. Each network entity added to a Discovery instance must have at least one regular identifier.
Group Identifier	A network entity identifier intentionally referring to potentially more than one instance. Multiple nodes of a particular network may be identically layered. So it makes sense to assign the same identifier to the same subset of functions in multiple nodes, although this need not be done. Thus, a group identifier may refer to multiple network entity instances. Examples of function subsets which can have group identifiers include APPN Network Node, OSI Intermediate System, PrintServer, etc. A network entity may have multiple group identifiers. A network entity may have both group identifiers and regular identifiers.
Universal Group Identifier	A group identifier beginning with X'00'. Universal group identifiers are architected. A network entity may have only one universal group identifier.
Server	An entity which is identified by a group identifier. Such entities are Heartbeated.
Group Find	A Find which is sent out to a group MAC address or a functional address. For example, a Find sent to the non-server functional address is a group Find.

6.1.2 Discovery Functional Overview

Primitive Requirements

The precise specification of the primitives used to invoke Discovery is outside the scope of this document. This document specifies only the formats and protocols necessary to ensure that different product implementations can inter-operate. However, this section describes the functions that must be offered by any acceptable set of primitives.

Adding network entity identifiers

Local network entities must be added to the DLC.LAN.MGR before they can be found by remote network entities. To add a network entity identifier means to inform a DLC.LAN.MGR that the entity is accessible through it. The DLC.LAN.MGR is given the entity's protocolId, group identifiers, if any, and regular identifier.

Finding network entities

To find a network entity, the discovery user provides the DLC.LAN.MGR with:

- its protocolId,
- either a regular identifier or a group identifier of the sought entity, depending upon whether a particular instance is sought or not and upon whether the sought entity has a group identifier or not,
- if the sought entity was identified by a group identifier, whether one or all discovered instances of the sought entity is to be returned to the user,
- whether one or all routes are to be returned for each returned instance of the sought entity,
- if only one route is to be returned for each returned instance of the sought entity, the minimum required frame size that all bridges along the route and both stations must support (this value may be 0 to indicate that any frame size is acceptable),
- whether Discovery may return cached information or not,
- and, if available, a MAC address of the sought network entity.

The user may specify that information on ALL discovered instances of the sought entity be returned. The user may also specify that ALL routes to each discovered instance be returned. Discovery returns to the user:

- a MAC address of each discovered instance of the sought entity, up to the maximum desired number of instances,
- an LSAP of each discovered instance of the sought entity, up to the maximum desired number (one or all those discovered) of instances,
- if Discovery is being used on a source-routing bridged LAN, a number of routes to each discovered instance of the sought entity, up to the maximum desired number of instances (one or all those discovered) and the maximum desired number (also one or all those discovered) of routes per instance; if just one is requested, the returned route must satisfy the minimum frame size constraint.

Discovery Functions

Caching: Stations may, at the implementer's option, cache any information received on Discovery frames.

The Rotary Function: A station which receives a Find frame need not send the resulting Found frame, if any, from the MAC address which received the Find. This flexibility is useful if a network entity is accessible through multiple adapters on the LAN network. In this case, it can open one adapter for receiving Finds and rotate the Found replies from the other adapters. This will disperse the connections among the available adapters. This operation is called the *rotary function*.

Suppose that a DLC.LAN.MGR manages multiple adapters and determines that two of them are on the same segment. It is recommended that the DLC.LAN.MGR in that case ensure that one or the other, but not both, Heartbeat any entity at any

time; similarly it is recommended that the DLC.LAN.MGR ensure that one or the other, but not both, send a Found for any entity in reply to any one Find.

One possible technique that the DLC.LAN.MGR can use to determine whether any two of its adapters are on the same segment is to send from one of the adapters a TEST command frame without a routing information field; this TEST command is addressed to the other adapter. If the sending adapter receives a TEST response frame, the two adapters are on the same segment. Otherwise, they are not. Another possible technique is to configure the DLC.LAN.MGR with the information. Implementers need not use either of these two techniques.

Migration Strategy

Migration from currently existent, Discovery-incapable stations to Discovery-capable stations is supported by third-party Founds. These frames are sent by a Discovery-capable station on behalf of a Discovery-incapable station, thus allowing entities at the latter station to be located by any other Discovery-capable station.

To facilitate the future addition of fields to Discovery frames, the Discovery architecture requires that an implementing station ignore any Discovery frame fields it does not recognize. Any architected fields added to Discovery frames in the future shall be optional and shall be added at the ends of the current frames.

6.1.3 Discovery Specifications

Qualification of Identifiers: A single LAN may support multiple higher-layer protocols. ProtocolId is included in some Discovery frames. It identifies the protocol suite (for example, SNA, TCP/IP, OSI) performed by the entity.

Discovery Functional Addresses and LSAPs

When running on a Token-Ring adapter, the Discovery protocol uses the following functional addresses:

- X'C000 0000 0004', the Heartbeat functional address,
- X'C000 0000 0040', the non-server functional address,
- X'C000 0001 0000', the server functional address.

The origin and destination LSAP for all Discovery frames is X'DC'.

When running on an Ethernet or an 802.3 network, Discovery uses the three group addresses with values equal to the above three functional addresses. The purpose of each group address is that of the equal functional address.

Timers and counters

Discovery includes the following timers and try counts:

- Find Timer (settable to 0.5, 1, 1.5, ..., 20 and defaulting to 9 sec). The expiration of this timer causes a Find frame to be repeated.
- Find Try Count (default = 2, max = 6). This value is the maximum number of times that a Find frame is sent.
- Heartbeat Repetition Timer (settable to the values 0.5, 1, 1.5, ... , 60 seconds and defaulting to 5 seconds). A Heartbeat is sent when the Heartbeat Repetition Timer expires if an unanswered HeartbeatRequest is outstanding. This timer is reset and started upon the transmission of a Heartbeat.
- Heartbeat Timer (settable to 1, 2, 3, ... , 120 seconds and defaulting to 9 seconds). The period of this timer determines how long a station waits for a

Heartbeat. This timer is reset and started upon the transmission of a HeartbeatRequest.

- HeartbeatRequest Try Count (default = 2, max = 6). This value is the maximum number of times that a HeartbeatRequest frame is sent.
- Initial Heartbeat Count (settable to the values 0, 1, 2, ..., 1000 and defaulting to 60). This is the number of unsolicited Heartbeats sent by a station on behalf of a server if the station becomes aware that the server has just started up, the server or the station itself has just recovered, or recovery from network partition has just occurred.

These values can be overridden by the LAN administrator. Also, a LAN manager, if present, has the ability to get and set all of these timers and counts via the Resource Manager component of the DLC.LAN.MGR.

6.1.4 Discovery Frames

There are four discovery frames: HeartbeatRequest, Heartbeat, Find, and Found. A station single-route broadcasts a HeartbeatRequest to the server functional address to cause recipients to respond with Heartbeats. If an identifier of the sought entity was previously added to a recipient and if the sought entity is a server, the recipient single-route broadcasts a Heartbeat to the Heartbeat functional address. The Heartbeat announces a MAC address of the sought entity, among other information, to all stations whose Heartbeat functional addresses are open. A Find causes each recipient to respond with an all-routes broadcast Found if an identifier of the sought entity has been added to it. These frames and the protocol which determines when they are sent are illustrated in Figure 6-5 on page 6-12 and Figure 6-6 on page 6-14.

The next figures provide an overview of the Discovery frames sent and the opening and closing of the Heartbeat functional address for different discovery invocations and for different values of the Heartbeat Repetition Timer at the time the HeartbeatRequest frame is received.

Figure 6-2 on page 6-7, Figure 6-3 on page 6-7, and Figure 6-4 on page 6-8 provide an overview of the frames sent in the execution of the Discovery protocol and the actions taken by the Discovery invoker's station and the sought entity's station. The central column labelled FRAME lists the frames sent, in sequence where time increases downwardly. The left-hand column labelled ACTION lists the actions taken by the Discovery invoker's station. The right-hand column labelled ACTION lists the actions taken by the sought entity's station. The caption of each figure indicates whether the sought entity is a server or a non-server.

Figure 6-2 on page 6-7 illustrates the case in which the Discovery invoker specifies a network entity identifier of a server. Here the Heartbeat Repetition Timer's value is non-zero when the HeartbeatRequest is received.

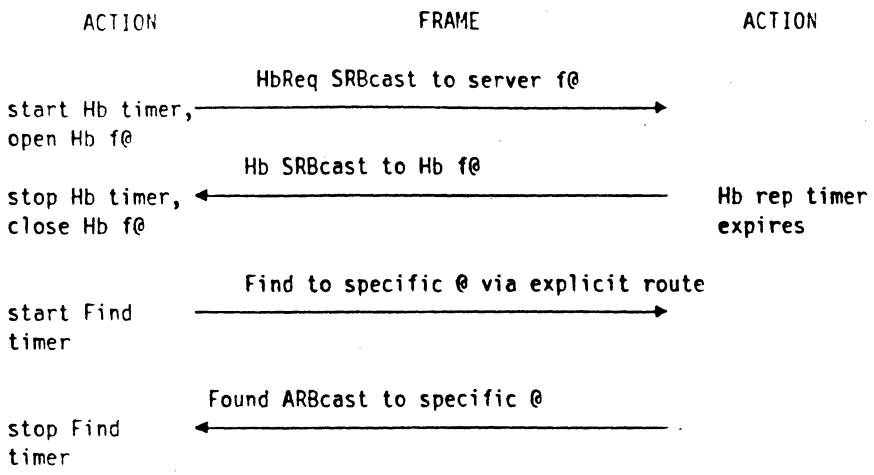
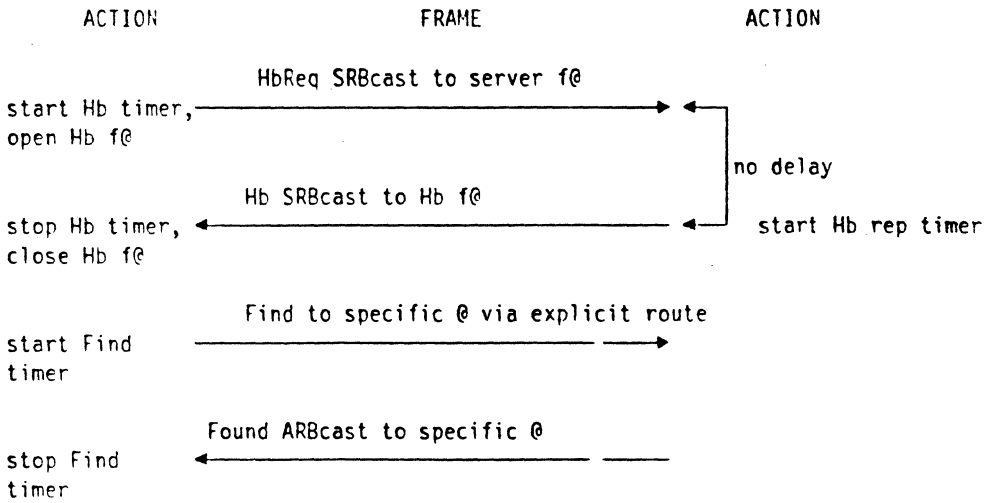


Figure 6-2. Finding a Server - Hb Repetition Timer Value is Initially Non-zero

The case illustrated by Figure 6-3 is the same as that by Figure 6-2 except that the Heartbeat Repetition Timer's value is zero when the HeartbeatRequest is received.



This protocol is performed whether or not the Discovery invoker knows that the sought entity is a server.

Figure 6-3. Finding a Server - Hb Repetition Timer Value is Initially Zero

Figure 6-4 on page 6-8 illustrates the case in which the Discovery invoker specifies a network entity identifier of a non-server and also specifies that the entity is a non-server.

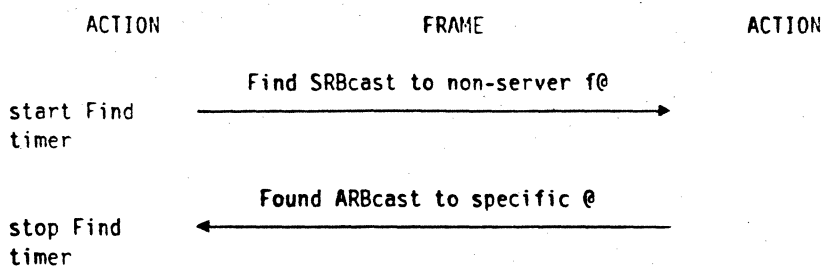


Figure 6-4. Finding a Non-server

HeartbeatRequest Frame Usage: If a station is invoked to locate a server, the station may single-route broadcast a HeartbeatRequest frame to the server functional address. Alternatively, if a station is invoked by a Discovery primitive which requests that an entity (whether a server or not) with its own architected functional address (such as LAN manager) be discovered, the station may simply single-route broadcast a Find to that functional address address. Doing so reduces the number of unnecessary interrupts.

If, after the transmission of a HeartbeatRequest frame for the sought entity, the Heartbeat Timer expires before a Heartbeat arrives, the station repeats the HeartbeatRequest frame. At most, it is sent a number of times equal to the Heartbeat Try Count.

If a station to which the sought server has added its identifier receives a HeartbeatRequest, it executes the following steps. If the Heartbeat Repetition Timer's value is 0, the station immediately single-route broadcasts a Heartbeat to the Heartbeat functional address. On the other hand, if the timer's value is non-zero, the station waits until the timer expires and then broadcasts the Heartbeat. If the station receives multiple HeartbeatRequests seeking the server before the Heartbeat Repetition Timer expires, it ignores all but one. In either case, after transmitting the Heartbeat, the station resets and starts the Heartbeat Repetition Timer.

HeartbeatRequest Frame Fields:

origRegularIdentifiers	This is a sequence of all regular identifiers which identify the entity invoking Discovery. It is present in all HeartbeatRequests.
searchGroupIdentifier	This is a group identifier of the entity being sought. A HeartbeatRequest frame must contain a group identifier or a regular identifier.
searchRegularIdentifier	This is a regular identifier of the entity being sought. A HeartbeatRequest frame must contain a group identifier or a regular identifier.

Heartbeat Frame Usage: If a station receives a HeartbeatRequest for a server whose identifier has been added to it, the station single-route broadcasts a Heartbeat to the Heartbeat functional address, after possibly waiting until the Heartbeat Repetition Timer expires, as described above. Upon start-up or

recovery, a station to which the identifiers of one or more servers have been added sends a sequence of Heartbeats of number equal to the Initial Heartbeat Count spaced in time by the Heartbeat Repetition Timer period for each such server. It is recommended that the station intersperse the sequences of Heartbeats for different servers. During the time that the station is performing this initial Heartbeating, it may keep its server functional address closed.

Heartbeat Frame Fields:

protocolId	The protocolId specifies the protocol performed by the Heartbeated entity. This field is present in all Heartbeats.
returnMacAddress	The returnMacAddress is a MAC address through which the server is accessible. It is the MAC address to which a Find frame to the server is addressed. This field is present in all Heartbeats.
groupIdentifiers	This is a sequence of all the group identifiers, if any, which identify the server.
regularIdentifiers	This is a sequence of all the regular identifiers which identify the server. At least one must appear. This field is present in all Heartbeats.

Find

Find Frame Usage: The main content of a Find frame is a regular or group identifier of the sought network entity and a MAC address and an LSAP of the invoking network entity. If a station is invoked by a Discovery primitive which requests that an entity be discovered and specifies that the entity is a non-server, that station simply single-route broadcasts a Find to the non-server functional address. Alternatively, if a station is invoked by a Discovery primitive which requests that an entity with its own architected functional address (such as LAN manager) be discovered, the station may simply single-route broadcast a Find to that functional address. This is true whether the entity is specified to be a non-server or not. Doing so reduces the number of unnecessary interrupts. If a station receives a Heartbeat for the sought entity, it responds with a Find sent to the specific MAC address contained in the Heartbeat. The Find follows the same route that the Heartbeat traversed.

If a Find results in no Found before the Find timer expires, the Find frame is repeated. It is sent at most a number of times equal to the Find Try Count. If a Find results in a Found-in-progress (defined below), it is retransmitted in this way, but over the explicit route that the Found-in-progress followed.

For a source routing bridged LAN, if the station does not know a regular identifier of the sought entity, but does know one of its MAC addresses and one of its LSAPs, the station may single-route broadcast or all-routes broadcast, at the implementer's option, a TEST (COMMAND) or an XID (COMMAND), again at the implementer' option, to the MAC address in order to discover the route. See *IBM Token Ring Architecture Reference* for a description of the use of TEST and XID frames.

Find Frame Fields:

correlator	All Find frames include a correlator field. The correlators of all Finds caused by the same Discovery primitive invocation are equal. Each Found frames contains the correlator of the Find to which it is replying.
protocolId	The protocolId specifies the protocol performed by the network entity invoking Discovery. This field is present in all Finds.
returnMacAddress	The returnMacAddress is a MAC address through which the entity invoking Discovery is accessible. It is the MAC address that the resulting Found is sent to and the MAC address through which communication subsequent to the Discovery exchange will go. This field is present in all Finds.
returnLSAP	The returnLSAP is an LSAP of the entity invoking Discovery. It is the LSAP through which communication subsequent to the Discovery exchange will go. This field is present in all Finds.
frameSize	A Find includes a field called FrameSize. It is set by the sending station to the size in bytes of the largest frame the station can receive. This field is present in all Finds.
origRegularIdentifier	This is a regular identifier which identifies the entity invoking Discovery. It is present in all Finds.
groupSearchIdentifier	This is a group identifier of the entity being sought. A Find must contain one group identifier or one regular identifier.
regularSearchIdentifier	This is a regular identifier of the entity being sought. A Find must contain one group identifier or a regular identifier.
snapProtId	This is the 802.1A protocol identifier. It is present if the entity invoking discovery is accessible through LSAP X'AA' or if the entity invoking discovery is an Ethernet application.

Found

Found Frame Usage: A Found frame's ReplyCode field informs the Found recipient whether the sought entity specified in the Find was found or not and whether its DLC.LAN.MGR has sufficient resources to support the communication or not. The sought entity is considered found if the search identifier and protocolId contained in the Find match those of some entity located at the recipient station. The value and length of both the search identifier and the protocolId from the Find frame must equal the value and length of some identifier stored in the DLC.LAN.MGR for a match to occur.

A Found indicating that the sought entity was found and that the DLC.LAN.MGR may have sufficient resources is a *positive Found* and has reply code X'00'.

A *Found-in-progress* is denoted with a ReplyCode equal to X'01' and informs the recipient that the Found source needs more time to determine if the sought entity is available. It causes the recipient to wait for the wait interval specified in the

replyCode field before retransmitting the Find. A Found-in-progress is a type of third-party Found. Upon reception of a Found-in-progress, a station sets the Find timer to the value of the waitInterval subfield of the replyCode field. The number of Find retries is unaffected by the extension of the Find timeout. If a station receives a sequence of Founds-in-progress from the same source in reply to the same Find, it performs the actions described above for each (i.e. the Find timer is set to the waitInterval value from each Found-in-progress).

A Found indicating that the sought entity was found and that its DLC.LAN.MGR does not have sufficient resources for additional communication is a *busy Found* and has reply code X'02'.

A Found indicating that the sought entity was not found is a *negative Found* and has reply code X'03'.

The recipient of a non-group Find always replies with a Found frame. The recipient of group Find replies with a Found only if the Found is positive or is a Found-in-progress. Founds are all-routes broadcast only if they are positive and are not third-party. All other Founds are sent over the explicit route followed by the Find which solicited them, if that route is known; if not, they are single-route broadcast.

For any Find, any consequent Found is sent to the return MAC address contained in the Find frame. The Found frame contains a MAC address and an LSAP through which the network layer entity with whom the origin of the Find frame wishes to communicate is accessible. This MAC address and this LSAP will be used for the communication between the sought entity and the Discovery invoker after the completion of the Discovery protocol. As with all broadcast frames to specific (non-group) addresses, the route taken by each Found frame as it traverses a source routing bridged LAN is accumulated in the frame. In this way, the initiating station dynamically learns a MAC address and an LSAP of the destination network entity, and, for a source-routing-bridged LAN, a route.

Found Frame Fields:

correlator	All Found frames contain the correlator of the Find which they are responding to.
returnMacAddress	The returnMacAddress is a MAC address through which the sought entity is accessible. For a non-third-party Found, this address is the same as the address of the adapter sending the Found. For a third-party Found, it is not. This field is present in all but negative Founds.
returnLSAP	The returnLSAP is an LSAP through which the sought entity is accessible. Note that it is not X'DC'. This field is present in all Founds except for Founds-in-progress and negative Founds.
replyCode	The replyCode field indicates whether the sought entity has been found or not, whether more time is required to search for it or not, and whether the DLC.LAN.MGR has sufficient resources to support the additional communication or not. If more time is needed, the waitInterval subfield of ReplyCode indicates how much is needed. This field is present in all Founds.

regularIdentifiers	This is one regular identifiers of the sought entity. It is present if the Find did not contain a regularSearchIdentifier. The regular identifier may be useful to the Found destination in deciding which replying network entity instance to communicate with.
frameSize	The frameSize field is set by the sending station to the size in bytes of the largest frame the station can receive. The Found recipient calculates the size of the largest frame it can send to the Found source to be the minimum of the contents of this frameSize field and the frame size indicated in the Routing Information field of the MAC header. This field is present in positive or busy non-third-party Founds.
snapProtId	This is the 802.1A protocol identifier. It is present if the sought entity is an Ethernet application or if the LSAP that this entity is accessible through is X'AA'.

6.1.5 Sample Discovery Flows

The following figures specify in detail the contents, sources, and destinations of Discovery frames.

First, Figure 6-5 illustrates the Discovery protocol exec /ted in the case that the sought entity is a server.

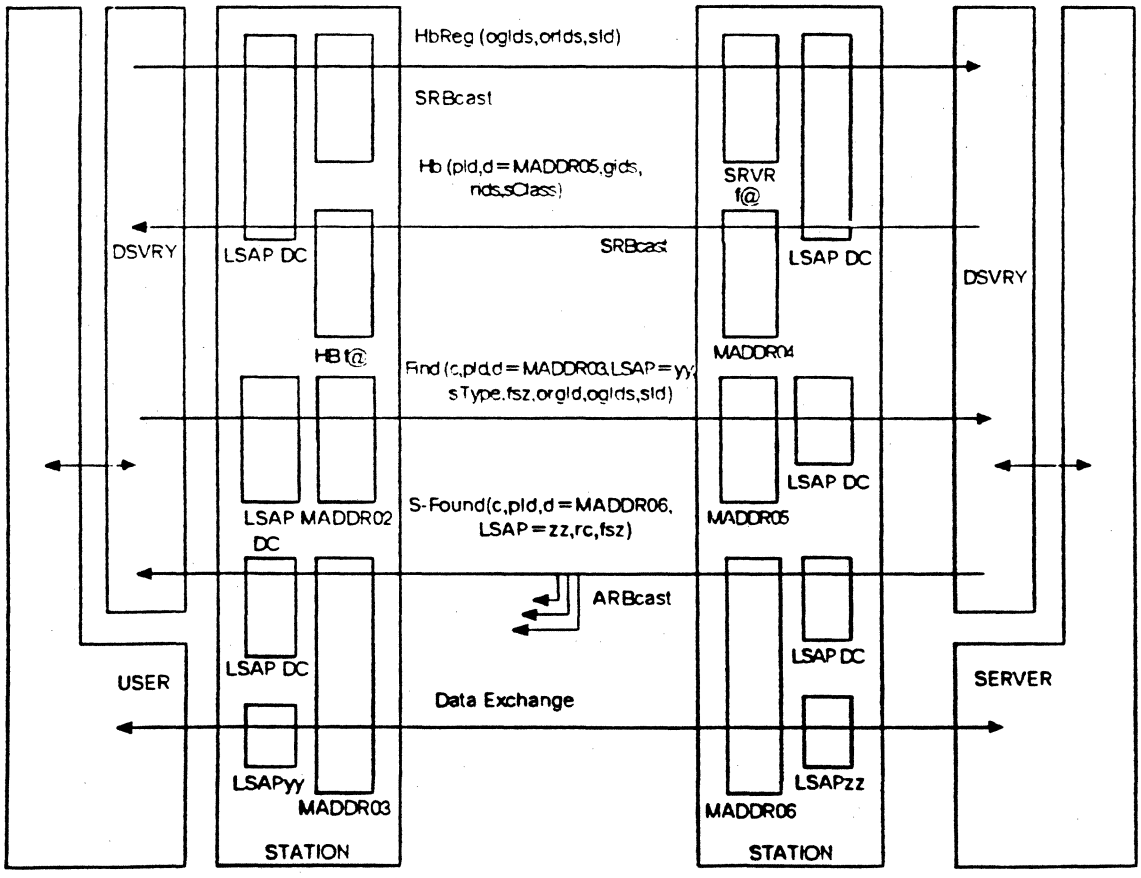


Figure 6-5. Server Discovery

Legend:

Key to Frame Parameters:

HbReq (HeartbeatRequest):
- ogIds: all group identifiers of the entity invoking Discovery
- orIds: all regular identifiers of the entity invoking Discovery
- sId: a regular or group identifier of the sought entity

Hb (Heartbeat):
- pId: protocolId
- d: destination MAC address to be used by Find frame
- gIds: all group identifiers of entity being Heartbeated
- rIds: all regular identifiers of entity being Heartbeated
- sClass: serviceClass

Find:
- c: correlator to pair Finds with consequent Founds
- pId: protocolId
- d: MAC address to be used for communication at origin
- LSAP: LSAP to be used for communication at origin
- sType: serviceType
- fSz: frameSize
- orgId: a regular identifier of the entity invoking Discovery
- ogIds: all group identifiers of the entity invoking Discovery
- sId: a regular or group identifier of the sought entity

Found:
- c: Correlator of the Find which initiated this Found
- pId: protocolId
- d: MAC address to be used for communication
- LSAP: LSAP to be used for communication
- rc: replyCode
- fSz: frameSize

Figure 6-6 on page 6-14 illustrates the Discovery protocol executed in the case that the sought entity is a non-server.

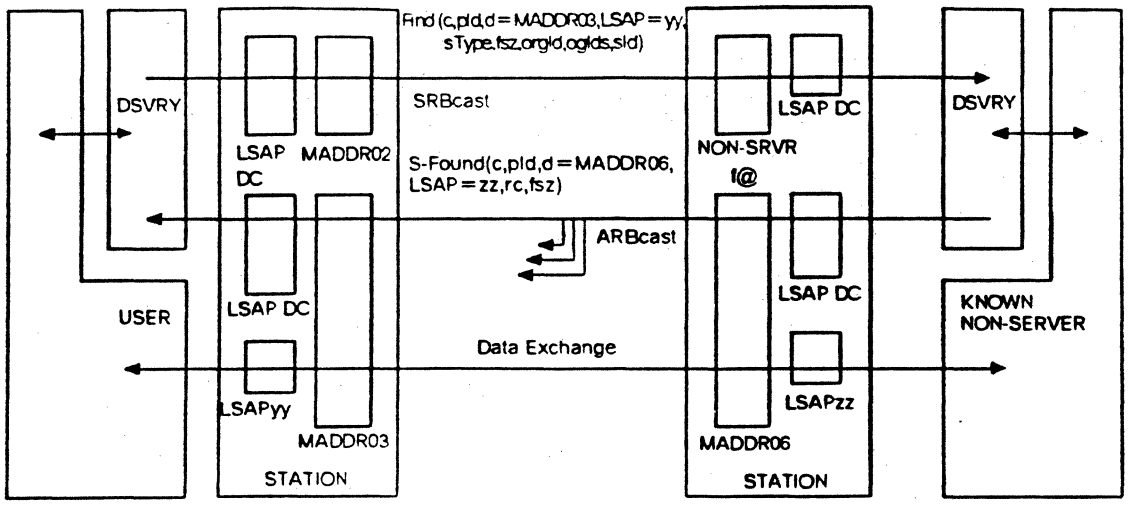


Figure 6-6. Non-Server Discovery

Legend: *Key to Frame Parameters:*

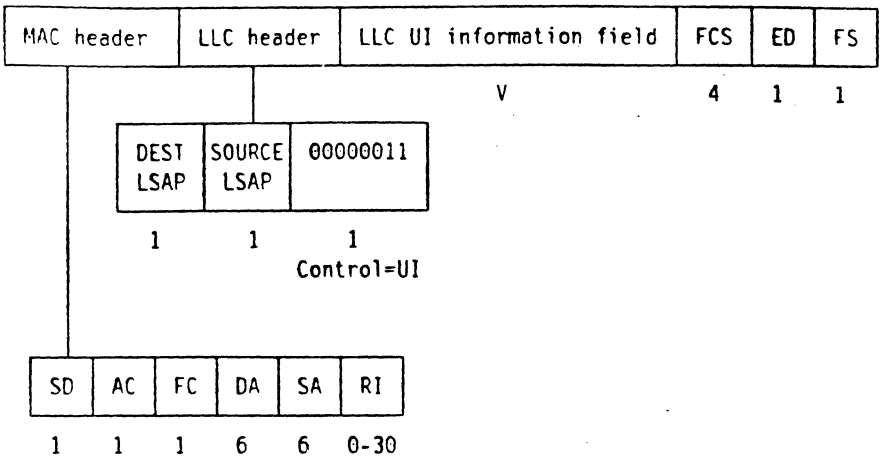
Find:
- c: correlator to pair Finds with consequent Founds
- pId: protocolId
- d: MAC address to be used for communication at origin
- LSAP: LSAP to be used for communication at origin
- sType: serviceType
- fSz: frameSize
- orgId: a regular identifier of the entity invoking Discovery
- ogIds: all group identifiers of the entity invoking Discovery
- sId: a regular or group identifier of the sought entity

Found:
- c: Correlator of the Find which initiated this Found
- pId: protocolId
- d: MAC address to be used for communication
- LSAP: LSAP to be used for communication
- rc: replyCode
- fSz: frameSize

6.1.6 Frame Formats

This subsection provides in ASN.1 the formats of the information field of the UI LLC frames of the Discovery function. The ASN.1 language is defined in ISO 8825:1987(E) with DAD1.

The following figure illustrates the position of the information field in the LLC frame for a token-ring LAN with source routing.



The total length of the frame must be less than or equal to 512 bytes.

Figure 6-7. Token-Ring Discovery Protocol Frame Format

The following legend explains some of the abbreviations of the above figure. See *IBM Token Ring Architecture Reference* for details concerning these fields.

SD Starting delimiter

AC Access control field

FC Frame control field

DA Destination address

HeartbeatRequest	Server functional address (X'C000 0001 0000'),
Heartbeat	Heartbeat functional address (X'C000 0000 0004')
Find	Non-server functional address (X'C000 0000 0040') or specific address
Found	Specific address

SA Source address (specific)

bit(0) = 0	Routing Information field not present
bit(0) = 1	Routing Information field present

RI Routing Information field

byte(0-1)	Routing control field	
bit(0) = 0	follow route in RI field (non-broadcast)	
bit(0) = 1	broadcast all rings (build RI field enroute)	
	bit(1) = 0	through all bridges
	bit(1) = 1	through only single-route broadcast bridges

FCS Frame check sequence

ED Ending delimiter

FS Frame Status field

Abstract Syntax Notation

Four UI LLC frames are used: HeartbeatRequest, Heartbeat, Find, and Found.

```
Discovery DEFINITIONS ::= BEGIN

DiscoveryFrame ::= CHOICE {
    heartbeatRequest [4] IMPLICIT HeartbeatRequest,
    heartbeatSequence [1] IMPLICIT HeartbeatSequence,
    find [2] IMPLICIT Find,
    foundSequence [3] IMPLICIT FoundSequence }

HeartbeatSequence ::= SEQUENCE OF Heartbeat


HeartbeatRequest ::= SEQUENCE {
    origRegularIdentifiers RegularIdentifiers,
    -- all identifiers of the entity invoking Discovery must be present
    searchGroupIdentifier [14] IMPLICIT GroupIdentifier OPTIONAL,
    -- either one searchGroupIdentifier or one searchRegularIdentifier must be present
    searchRegularIdentifier [2] IMPLICIT RegularIdentifier OPTIONAL,
    -- either one searchGroupIdentifier or one searchRegularIdentifier must be present

Heartbeat ::= SEQUENCE {
    protocolId ProtocolId,
    returnMacAddress [3] IMPLICIT MacAddress,
    groupIdentifiers [9] IMPLICIT GroupIdentifiers OPTIONAL,
    -- all group identifiers of the server must be present
    regularIdentifiers [2] IMPLICIT RegularIdentifiers,
    -- all regular identifiers of the server must be present

Find ::= SEQUENCE {
    correlator Correlator,
    protocolId ProtocolId,
    returnMacAddress [3] IMPLICIT MacAddress,
    returnLsap [5] IMPLICIT Lsap,
    frameSize [7] IMPLICIT FrameSize,
    origRegularIdentifier RegularIdentifier OPTIONAL,
    -- a regular identifier of the entity invoking Discovery must be present
    --    in a Find to a known non-server or in a Find destined for the LAN
    --    manager functional address
    searchGroupIdentifier [9] IMPLICIT GroupIdentifier OPTIONAL,
    -- either one search group identifier or one search regular identifier must be present
    searchRegularIdentifier [2] IMPLICIT RegularIdentifier OPTIONAL,
    -- either one search group identifier or one search regular identifier must be present
    snapProtId [13] IMPLICIT SnapProtId OPTIONAL,
    -- present if the LSAP the sought entity is accessible through is X'AA'
    --    or if the entity invoking Discovery is an Ethernet application

FoundSequence ::= SEQUENCE OF Found

Found ::= SEQUENCE {
    correlator Correlator,
    returnMacAddress [3] IMPLICIT MacAddress,
    returnLsap [5] IMPLICIT Lsap,
    -- present except in negative Founds or Founds-in-progress
```

```
    replyCode ReplyCode,
    regularIdentifier [2] IMPLICIT RegularIdentifier OPTIONAL,
    -- required if the Find did not include regularSearchIdentifier
    stationType [6] IMPLICIT StationType OPTIONAL,
    -- present in a third-party Found
    frameSize [7] IMPLICIT FrameSize OPTIONAL,
    -- present in a non-third-party Found that is positive or busy.
    snapProtId [13] IMPLICIT SnapProtId OPTIONAL,
    -- present if the LSAP the sought entity is accessible through is X'AA'
    --    or if the sought entity is an Ethernet application

Correlator ::= OCTET STRING (4)

FrameSize ::= OCTET STRING (4)

GroupIdentifiers ::= SEQUENCE OF GroupIdentifier

RegularIdentifiers ::= SEQUENCE OF RegularIdentifier

RegularIdentifier ::= OCTET STRING

GroupIdentifier ::= OCTET STRING

Lsap ::= OCTET STRING (1)

MacAddress ::= OCTET STRING (6)

ProtocolId ::= OCTET STRING (2)

ReplyCode ::= OCTET STRING (0-1)

StationType ::= OCTET STRING (1)

SnapProtId ::= OCTET STRING (3)


END
```

Primitive Data Types:

Correlator
- value octets 0–3: correlator

FrameSize
- value octets 0–3: the maximum frame length supported by the frame source station

GroupIdentifier
- value octets 0–n: a group identifier. those group identifiers beginning with X'00' are universal group identifiers and are reserved.
  - X'0000' LAN manager
  - X'0001' Controlled access unit (CAU)

Lsap
- value octet 0: Link Services Access Point

MacAddress
- value octets 0–5: MAC Address in canonical form

ProtocolId
- value octets 0–1: protocolIdentifier
  - b'1xxxxxxx xxxxxxxx' Locally administered identifier
  - b'00000000 00000000' Systems Network Architecture - subarea
  - b'00000000 00000001' Systems Network Architecture - APPN
  - b'00000000 00000011' Transmission Control Protocol/ Internet Protocol (TCP/IP).
  - b'00000000 00000100' Netbios
  - b'00000000 00000101' DECNET
  - b'00000000 00000110' CMIP
  - all others reserved

RegularIdentifier
- value octets 0–n: regular identifier

ReplyCode
- value octet 0: reply code
  - X'00' positive Found - the sought entity has been found and the sending DLC.LAN.MGR currently may have sufficient resources for additional communication.
  - X'01' Found-in-progress - the Found source needs more time to determine if the sought entity is available. Wait for the time interval specified in octet 1 before sending another Find. This is a Found-in-progress.
  - X'02' busy Found - the sought entity is available but the sending DLC.LAN.MGR does not have sufficient resources for additional communication.
  - X'03' The sought entity was not found. Sent only in reply to a Find to a specific MAC address.
  - others reserved
- value octet 1: Wait interval
  If byte 2 has value X'01', this field contains the length of time in seconds that the receiving station should wait before retransmitting a Find. Otherwise, it has the value X'00'.

StationType
- value octet 0:
  - X'00' the station is Discovery-incapable
  - X'01' the station is Discovery-capable
  - others reserved

SnapProtId
- value octets 0–4: SNAP protocol identifier, defined in IEEE 802.1A

6.1.7 Discovery Finite State Machines

Finite State Machine 1

FSM 1 describes the actions of a station using Discovery to locate one instance of a remote entity whose identifier, but not MAC address or LSAP, is known. It is assumed that the identifier has been added to another (Discovery-capable) station. It is assumed that the sought entity does not have its own architected functional address. It is assumed that both stations are located on the same bridged token-ring LAN with source routing. It is also assumed that the stations through which the sought entity is accessible have been operating long enough that they have stopped their initial Heartbeating.

For the Discovery FSMs, resetting a timer means setting its value to its period. That is, the FSM's assume that timers count down. Nevertheless, product implementers may implement Discovery timers that count up. Stopping a timer means causing its value to remain constant until it is reset. It is assumed that timers stop after expiring.

It is assumed that each Discovery invocation begins the execution of a different instance of FSM 1. Therefore, an already executing instance of FSM 1 cannot receive a Discovery invocation; that is, it cannot receive a Locate A Server, a Locate A Non-server, or a Locate input.

States:

- Reset: The station is waiting for a Discovery invocation.
- Listen: The station is listening for a Heartbeat or a Found. Whether the sought entity is a server or not is known.
- Server?: The station is listening for a Heartbeat. Whether the sought is a server or not is unknown.

Inputs

Input Name	Description
Locate a Server.	A station receives a Discovery invocation which includes a network entity identifier of a server. Discovery is to determine a MAC address and LSAP of one instance of this entity and find a route to it.
Locate a Non-server.	A station receives a Discovery invocation which includes a network entity identifier of a non-server. Discovery is to determine a MAC address and LSAP of one instance of this entity and find a route to it.
HB Arrives.	A Heartbeat arrives.
Hb Timer Expires, retries not Exh.	A Heartbeat containing the identifier of the sought network entity is not received before the Heartbeat listen timer expires. Further, the Heartbeat Request has not been sent a number of times equal to the HeartbeatRequest Try Count.
Hb Timer Expires, retries Exh.	A Heartbeat containing the identifier of the sought network entity is not received before the Heartbeat listen timer expires. Further, the Heartbeat Request HAS been sent a number of times equal to the HeartbeatRequest Try Count.
Found(p or b) Arrives.	A positive or busy Found with correct correlator value arrives. The "p" stands for positive and the "b" stands for busy.
Found(n) Arrives.	A negative Found with correct correlator value arrives. The "n" stands for negative.
Found-in-progress Arrives.	A Found-in-progress with correct correlator value arrives.
Find Timer Expires, retries not Exh.	The Find Timer expires before the arrival of a positive, busy, or negative Found with correct correlator value. Additionally, the Find frame has not already been sent a number of times equal to the Find Try Count.
Find Timer Expires, retries Exh.	Same as the previous item, except that the Find frame HAS been sent a number of times equal to the Find Try Count.
Deactivate arrives.	The signal which stops the execution of the Discovery protocol arrives.

FSM 1

Inputs	States	
	Reset	Listen
	01	02
Locate a Server.	2(d)	/
Locate a Non-server.	2(i)	/
HB Arrives	/	-(b)
Hb Timer Expires, retries not Exh.	/	-(j)
Hb Timer Expires, retries Exh	/	1(f)
Found(p or b) Arrives.	/	1(e)
Found(n) Arrives.	/	1(f)
Found-in-progress Arrives.	/	-(h)
Find Timer Expires, retries not Exh.	/	-(c)
Find Timer Expires, retries Exh.	/	1(f)
Deactivate Arrives.	-	1(g)

Outputs:

Output Code	Function
a	Single-route broadcast Find to Locate Functional Address. Reset and start Find Timer. Reset Find Try Count.
b	Single-route broadcast Find to return MAC address. Reset and start Find Timer. Reset Find Try Counter to Find Try Count - 1. Stop Heartbeat Timer. Close Heartbeat functional address.
c	Retransmit previous Find. Reset and start Find timer. Decrement Find Try Counter.
d	Single-route broadcast HeartbeatRequest to server functional address. Reset and start Heartbeat Timer. Open Heartbeat functional address. Reset HeartbeatRequest Try Counter.
e	Signal that the sought network entity has been located. Stop Find timer.
f	Signal that the sought network entity has not been located. Stop Heartbeat Timer and Find timer, if running.
g	Signal that deactivation has occurred. Stop Heartbeat Timer and Find Timer, if running. Close Heartbeat functional address, if open.
h	Reset Find Timer to Wait Interval from replyCode and start it.
i	Single-route broadcast Find to Non-server Functional Address. Reset and start Find Timer. Reset Find Try Counter to Find Try Count - 1.
j	Single-route broadcast HeartbeatRequest to server functional address. Reset and start Heartbeat Timer. Open Heartbeat functional address. Decrement HeartbeatRequest Try Counter.
-	No state change
/	This input cannot occur in this state

Finite State Machine 2

FSM 2 describes the actions of a station receiving a HeartbeatRequest frame for a server whose identifier has been added to the station. It is assumed that one instance of FSM 2 is running for each such server. It is also assumed that the station has been operating long enough that it has stopped its initial Heartbeating.

States:

- NoHbR=0: No HeartbeatRequest has been received since the transmission of the last Heartbeat and the Heartbeat Repetition Timer's value is 0.
- NoHbR>0: No HeartbeatRequest has been received since the transmission of the last Heartbeat and the Heartbeat Repetition Timer's value not 0.
- HbR>0: At least one HeartbeatRequest has been received since the transmission of the last Heartbeat and the Heartbeat Repetition Timer's value not 0.

Inputs:

Input Name	Description
HeartbeatRequest Arrives.	A station receives a HeartbeatRequest for a server which is included in the DLC.LAN.MGR's list of servers for which it must Heartbeat.
Heartbeat Repetition Timer Expires.	The Heartbeat Repetition Timer expires.

FSM 2

	States		
	NoHbR=0	NoHbR>0	HbR>0
Inputs	01	02	03
HeartbeatRequest Arrives	2(a)	3	-
Heartbeat Repetition Timer Expires	/	1	2(a)

Outputs:

Output Code	Function
a	Single-route broadcast a Heartbeat to the Heartbeat functional address. Reset and start the Heartbeat Repetition Timer.
-	No state change.
/	This input cannot occur in this state.

FSM 3 describes the actions of a station receiving a Find frame for an entity whose identifier has been added to it. It is assumed that one instance of FSM 3 is running in each station.

States:

- Listen: The DLC.LAN.MGR is listening for Finds.

Inputs:

Inputs	Description
Non-group Find to Loc Reg Entity Arrives.	A station receives a non-group Find to a entity which is included in the DLC.LAN.MGR's list of local entities.
Group Find to Loc Reg Entity Arrives.	A station receives a group Find to a entity which is included in the DLC.LAN.MGR's list of local entities.
Non-group Find to Fip Entity Arrives.	A station receives a non-group Find to an entity which is included in the DLC.LAN.MGR's list of remote entities for which it mus t send Founds-in-progress.
Non-group Find to Unregis-tered Entity Arrives.	A station receives a non-group Find to an entity which is not included in either of the DLC.LAN.MGR's two lists of entities.

FSM 3:

	States
	Listen
Inputs	01
Non-group Find to Loc Reg Entity Arrives.	-(a)
Group Find to Loc Reg Entity Arrives.	-(a)
Find to Fip Entity Arrives.	-(b)
Non-group Find to Unregistered Entity Arrives.	-(c)

Outputs:

Output Code	Function
a	All-routes broadcast positive Found to ReturnMACAddress from Find.
b	Send via the explicit route from the Find, if available, otherwise Single-route broadcast Found-in-progress to ReturnMACAddress from Find.
c	Send via the explicit route from the Find, if available, otherwise Single-route broadcast negative Found to ReturnMACAddress from Find.
-	No state change.
/	This input cannot occur in this state.

Chapter 7. Security

7.1 Security

Security for management in a distributed environment is of the utmost concern, and has in the past been the limiting factor in the success of other management protocols. Since this architecture has adopted the work done by ISO, and its CMIP definitions, it has not ruled out security. However, definition for the use of the access control field, as defined in the CMIP frames, has recently begun in the OSI Management Standards group (SC21 xxxx), and is not at this time complete. The progress of this group will be monitored, and when the work has reached International Standards status, it will be reviewed for adoption by this architecture. In the interim, it is in the interest of this document to define an acceptable mechanism for implementing Security.

Chapter 8. CMIP Event

The CMIP EVENT is used for an unsolicited report of events occurring at the LAN entity. The event report can be either a confirmed or an unconfirmed type. The information contained in the Event Report may include the error data, performance data, configuration information, and other management application information.

After a LAN Station Manager is initialized, it will register with a Managing Process (as described in Chapter 17, "Registration" on page 17-1). A Managing Process with which it registers will be known as a registered Managing Process. The LAN Station Manager will send all event reports to its registered Managing Process(es). If any GETs are received from a station that is not a registered Managing Process, the LAN Station Manager will send a General Report (Nonregistered GET) event report to notify its registered Managing Process(es). In addition, all SETs are reported to all registered Managing Processes (except the managing process that initiated the set by way of the General Report (Set Occurred).

To ensure that the Managing Process is still present and receiving the event reports, occasionally a confirmed event report will be sent. After a certain amount of time, the next event to be sent is sent as a confirmed event. This timer is configurable and can range from zero (all events confirmed) to infinity (no events confirmed).

Figure 8-1 illustrates the generic flow of the event report. The parenthesis contain the subfields. The event response flows only when the event report is a confirmed event.

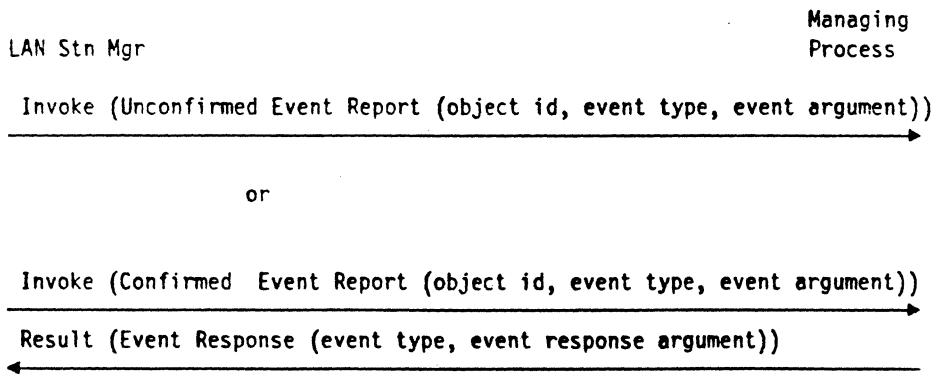


Figure 8-1. Management Flows for Event Report

Figure 8-2 on page 8-2 shows an example of the flow of events (unconfirmed and confirmed).

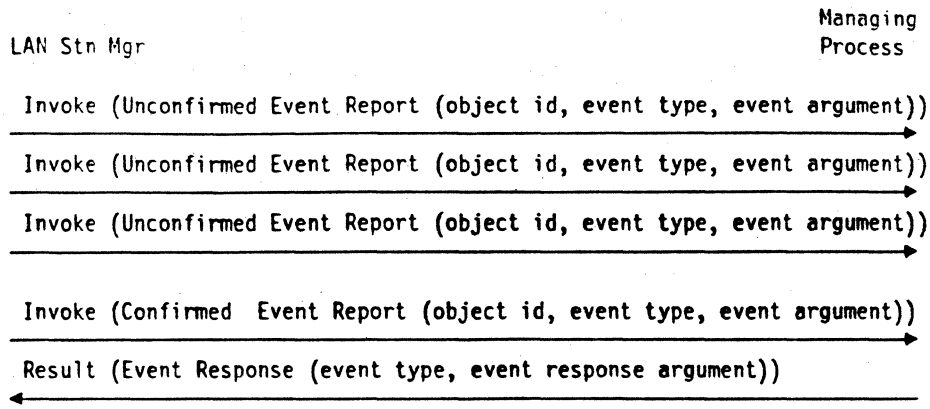


Figure 8-2. Management Flows for Event Report

The Event Report is used when a Layer Management Entity (LME) detects a fault condition or other activity in its corresponding layer or sublayer. The LME reports to the LAN Station Manager and the LAN Station Manager sends this information to the managing process through the CMIP Event Report.

8.1 General Format of the Event Report

CMIP Service	CMIP Parameters	Mgt Parameters
Event Report	Object ID	Layer Object
	Event Time	(Optional)
	Event Type	Alarm General Report Function Present Deregister Multiple Function Present Multiple Function Deregister
	Event Argument	e.g. Problem Data

Figure 8-3. Event Report and its associated parameters

There are four main fields for the Event Report: the Object Id, the Event time, Event Type and the Event Argument. These fields are discussed in this section.

Note: The objects and attributes are defined in OSI Management template format with accompanying ASN.1 definitions in 21.1, "Managed Object Templates" on page 21-1. The RO Invoke and the CMIP MPDU for the Event Report is defined in Chapter 16, "Protocol Data Unit ASN.1" on page 16-1.

8.1.1 Object ID

This field identifies the managed object for which the Event Report is generated. It is made up of two components : Managed Object Class and Managed Object Instance.

8.1.2 Event Time

This field contains the time at which the response was generated. It is optional.

8.1.3 Event Type

There are four types of Event Report:

- Alarm (See Chapter 22, "Alarm and Alert Definitions" on page 22-1) Should I refer them to Alarm and Alert definition chapter here?
- General Report
- Function Present (which reports that a function is present and is searching for a manager)
- Deregister (which reports that a function is deregistering from a manager.)
- Multiple Function Present (which reports that a group of functions are present and searching for a manager)
- Multiple Function Deregister (which reports that a group of functions are deregistering from a manager)

8.1.4 Event Argument

Event Types and Associated Data

The following tables illustrates the mapping of event type to the components of event argument.

Event Type	Event Argument			
	Description	Cause	Required Data	Reference
Alarm				
GeneralReport				
	LSAPGeneralEvent	LSAPOpened	LSAPid	8-5
			Supported Types	
			MaxIfield	
			MaxLinkStnsConfigured	
	LSAPPairGeneralEvent	LinkStationConnected	LLC Status 1	8-6
			Correlator 1	8-7
		LinkStationDisconnected	LLC Status 1	8-6
			Correlator 1	8-7
	Concentrator GeneralEvent	NewComAddress	MAC Address	8-6
		LobeStatusChange	Adapter Lobe Number	8-6
			Enable Status	
			Insert Status	
			MAC Address	
		AMStatusChange	AMStatus	8-6

Event Type	Event Argument			
	Description	Cause	Required Data	Reference
	MiscGeneralEvent			
		SETOccurred	Setter's MAC Address	8-6
			Attribute List	
		NonRegisteredGET	Getter's MAC Address	8-6
			Attribute List	
		DeviceOnline	[4]	8-6
		DeviceOffline	[4]	8-6
FunctionPresent	[2]	[3]	GroupFunctionTitle	8-7
			Qualifier	
			DistinguishingAttribute	
			PrimaryName	
			MACAddress	
			UserData	
Deregister	[2]	[3]	GroupFunctionTitle	8-7
			Qualifier	
			DistinguishingAttribute	
			PrimaryName	
			MACAddress	
			UserData	
Multiple Function Present	[2]	[3]	SET of Group Function information	8-8
			PrimaryName	
			MACAddress	
Multiple Function Deregister	[2]	[3]	SET of Group Function Information	8-8
			PrimaryName	
			MACAddress	

Alarm

For the definition of Alarms sent by the LAN Station Manager, see Chapter 22, "Alarm and Alert Definitions" on page 22-1

---

[1] Required Field Supplies Supporting Data although not the primary data field.

[2] The event argument for this event type has no description.

[3] The event argument for this event type has no cause.

[4] No data is sent with this event.

General Report

This section describes the event data for the general report.

Event data for the general report has the following subfields:

- General Description and Cause
- General Data (optional)
- LLC Status (optional)
- Correlator (optional)
- User data (optional).

In the General Report, only the General Description and Cause field differs from the Problem Report Event Data.

General Description And Cause

General description is a high level description of what layer general event has occurred:

- LSAP General Event
- LSAP Pair General Event
- Concentrator General Event
- Miscellaneous General Event.

The General Event Report Cause field adds detail to the reason for the General Report:

- LSAP Opened
- Link Station Connected
- Link Station Disconnected
- New Com Address
- Lobe Status Change
- Attachment Module Status Change
- SET occurred
- Non registered GET
- Device on-line
- Device off-line.

8.1.5 General Data

LSAP Opened

This notification is emitted when an LSAP is opened for sending data. The following data is sent.

- LSAPid
- SupportTypes
- Maximum LLC Information Field
- Maximum Link Stations Configured.

8.1.6 Link Station Connected Data

LLC status should be sent as the data to these event reports.

8.1.7 Link Station Disconnected Data

Upon termination of an LSAP Pair (regardless of the cause of termination), this general event is sent to to the managing processes. The LLC status and correlator are sent with this event report.

8.1.8 New Com Address Data

The data for a new com address is that new address.

8.1.9 Lobe Status Change Data

The data for a lobe status change is a set of the following information:

Adapter lobe number

Enable status - (activated or deactivated)

insert status - .(not inserted or inserted)

MAC address.

8.1.10 Attachment Module Status Change Data

The data that is sent with an Event Report initiated by attachment module status change is a set of the attachment module number and the status of that module (not present, active, deactivated).

8.1.11 SET Occurred SET

When a set is performed, a general event is sent to all managing processes except the one that requested the set. This general event has a cause of SET occurred. The following is sent as the data:

MAC address of setter
attribute list of what was set

8.1.12 Non registered GET Data

MAC address of getter
attribute list of what was gotten

8.1.13 Device on-line Data

No data is sent with this event, which is sent by the Resource Management object class.

8.1.14 Device off-line Data

No data is sent with this event, which is sent by the Resource Management object class.

LLC Status

The LLC Status contains information about link station connections. Its presence is required for LSAP Pair General events (except counter- related events.) It consists of the following elements:

- LSAP Pair ID
- k

- rW
- maximum retransmissions
- T2 timer timeout value
- T1 timer timeout value
- TI timer timeout value
- Max out increment
- Access priority
- Route (if available).

8.1.15 Correlator

The correlator contains a value that allows the management focal point to correlate problems if it receives multiple event reports from both local and remote nodes. It is always present in LSAPPair General Events. (See Chapter 9, "CMIP Action" on page 9-1 for details.)

8.1.16 User Data

The user data is an optional field that may contain any data the user wishes to send.

8.1.17 Function Present

The Function Present event is sent by the resource management object class when an application or a MAC and LLC attaches to the station manager. It contains the following data.

- group function title - which indicates the function that this entity provides.
- qualifier - indicates a division of the group (for example, the group of Token-Ring MAC will have a qualifier of segment number.)
- distinguishing attribute - the attribute which can be used to distinguish one function instance from another
- primary name - indicates the unique name of the managing agent registration.
- MAC address - indicates the MAC address through which the managing process can reach the entity.
- code pages
- architecture release level - the release level of the station manager
- user data - other information an entity may send

8.1.18 Deregister

The Deregister Event is sent to deregister an entity from a managing process. The following is the data contained in the Deregister event type:

- group function title
- qualifier
- distinguishing attribute
- primary name
- MAC address

- userdata.

8.1.19 Multiple Function Present

The following is the data contained in the MultipleFunctionPresent event type, which is sent when a station manager has multiple functions to announce.

- SET OF
  - GroupFunctionTitle, Qualifier, Distinguishing Attribute, Userdata
- primary name
- macAddress
- architecture release level.

Multiple Function Deregister

The following is the data contained in the MultipleFunctionDeregister event type, which is sent when a station manager has multiple functions to deregister. See 17.4.4, "Multiple Function Registration" on page 17-11 for details.

- SET OF
  - GroupFunctionTitle, Qualifier, Distinguishing Attribute, Userdata
- primary name
- macAddress.

8.2 Confirmed Event State Diagram

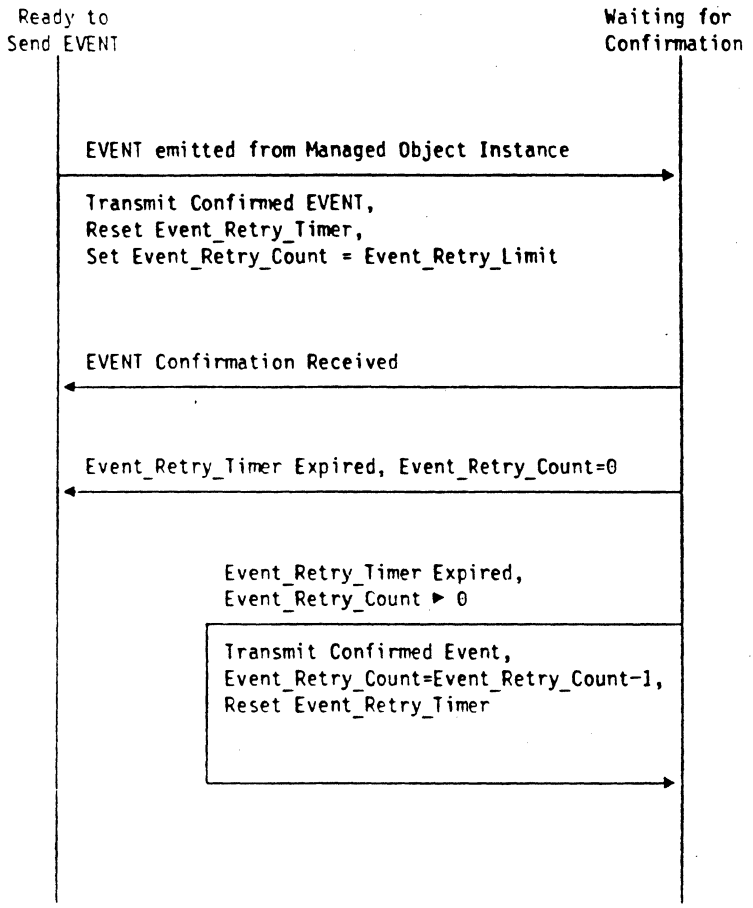


Figure 8-4 Confirmed Event State Diagram

Chapter 9. CMIP Action

This chapter describes the LAN Management functions that use the system management CMIP protocol verbs Unconfirmed and Confirmed ACTION. Figure 9-1 shows the generic flow of Action. The Actions will flow either between LAN Station Manager and a managing process or between LAN Station Managers. All actions except for Correlator Exchange are initiated by the managing process and sent to the LAN Station Manager. The Correlator Exchange is initiated by a station and sent to another station.

All actions except for RegisterRequest, MultipleRegisterRequest, RegisterCheck, and Correlator Exchange must come from a registered managing process. Otherwise the action is not performed and a response is returned with ActionError equal to access denied.

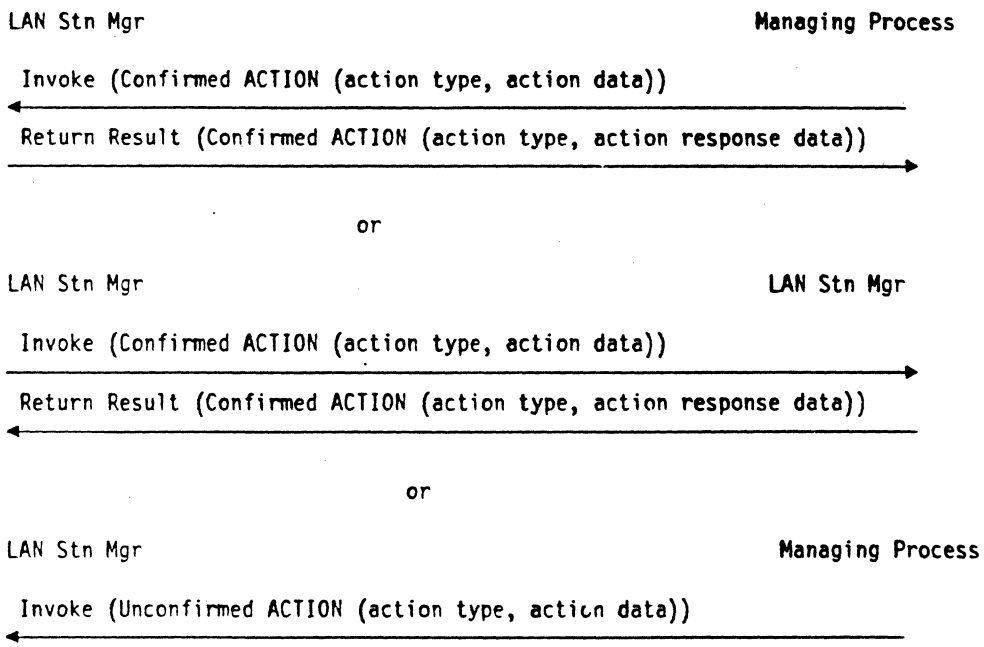


Figure 9-1. Management Flows for Actions

The following sections describe the Actions currently defined for LAN Management:

9.1 General Format of ACTION

CMIP Service	CMIP Parameters	Mgt Parameters
Action	Object ID	Object Class and Instance
	Action Type	Reinitialize Activate SAP Deactivate SAP Register Request Deregister Request Register Check Correlator Exchange Remove Station CAU Wrap Soft Reset RPU Enable Multiple Register Request Multiple Deregister
	Action Data	Depends on Action type

Figure 9-2. Confirmed Action and its associated parameters

There are three main fields for the Confirmed Action: the object Id, the Action Type and the Action Data. These fields are discussed in this section.

Note: The objects and attributes are defined in OSI Management template format with accompanying ASN.1 definitions in 21.1, "Managed Object Templates" on page 21-1. The RO Invoke and the CMIP MPDU for the ACTION is defined in Chapter 16, "Protocol Data Unit ASN.1" on page 16-1.

9.1.1 Object ID

This field identifies the managed object for which the Confirmed Action is destined. Managed Object Class and Managed Object Instance make up this field.

9.1.2 Action Type

The following are the Actions currently defined:

- Reinitialize
- Activate SAP
- Deactivate SAP
- Register Request
- Deregister Request
- RegisterCheck
- Correlator Exchange
- Remove Station
- CAU Wrap

- Soft Reset
- Remote Program Update (RPU) Enable
- Multiple Register Request
- Multiple Deregister.

Each of these actions are defined in the following sections:

Reinitialize

Reinitialize causes all connection to be disconnected, all LSAPs to be deactivated and the entire LLC sublayer to be reset to its initial configuration. No data is included in the action or its response. This action is required by the 802.2 layer management standard. It may be responded to negatively with access denied.

Activate SAP

Activate SAP causes the LSAP to be activated for Type 1, Type 2 or Type 3 operation or any combination of those types. The action data includes a list of the associated types to be activated. The response includes a list of the types which are active at the conclusion of this action. This action is required by the 802.2 layer management standard. It may be responded to negatively with access denied.

Deactivate SAP

Deactivate SAP causes the LSAP to be deactivated for Type 1, Type 2 or Type 3 operation or any combination of those types. The action data includes a list of the associated types to be deactivated. The response includes a list of the types of LLC operation which are active at the conclusion of this action. This action is required by the 802.2 layer management standard. It may be responded to negatively with access denied.

Register Request

Register Request is used by a managing process to request that a LAN Station Manager register an entity it is managing with the managing process. See Chapter 17, "Registration" on page 17-1 for details. The data includes the following information:

- group function title and qualifier of the entity it wishes to manage.
- distinguishing attribute for the objects in the group function title which provides distinction among several entities with the same group function title.
- primary name
- the MAC address through which this entity can be reached
- the managing process title
- the managing process address
- the managing process level
- the number of times the station manager should try to find a lost managing process
- security access field
- the sap the LAN Station Manager should use to communicate with this managing process.
- Confirmed Event Retry Timeout
- Confirmed Event Retry Number

- user data.

The response to the Register Request Action includes the following:

- group function title and qualifier
- distinguishing attribute
- primary name
- MAC Address
- arch release level
- return code
- security access field
- Character Set
- user data.

Deregister Request

Deregister Request is used by the managing process to request that a LAN Station Manager deregister itself with the managing process. The data includes:

- the group function title and qualifier of the entity
- distinguishing attribute for the group function title
- primary name of the resource manager
- MAC address through which this entity can be reached
- the managing process's title and address
- user data.

This action is unconfirmed.

Register Check

Register Check is used by a managing process to check the registration Status of an entity. See Chapter 17, "Registration" on page 17-1 for details. The data includes the following information:

- the managing process title and address
- group function title and qualifier of the entity it wishes to check the registration status
- the response type (all stations that are registered should respond, all stations that are NOT registered should respond, OR all stations with that group function title and qualifier should respond regardless or there registration status.)

The Station Manager returns the following data:

- group function title and qualifier
- distinguishing attribute
- primary name
- macAddress
- a boolean to indicate if the particular managing process requesting the register check is registered.

- a list of all registered managing processes for this group function title and qualifier.

Correlator Exchange

Correlation Exchange: Two ends of a link connection often report errors relating to same problem. A correlator is reported in the events so that the managing process can assume that the errors are related to the same failure.

This section describes the function that uses the CMIP protocol verb Confirmed Action to facilitate a correlator exchange between the two ends of a link connection. At the end of the link connection set-up (when the LSAP Pair has initially entered into LINK-OPENED state), the LAN Station Manager with the lowest MAC address will send a confirmed Correlator Exchange action to the LAN Station Manager of the adjacent link station. Upon receipt of this Action, the LAN Station Manager will compute a correlator. Once this correlator is generated, it is returned to the initiating Station Manager. This correlator will be sent in all events relating to the link connection.

If the response is not received, no correlator is sent in the events.

If the MAC address of the two ends of the link connection are the same, the link station that initiated the connection (sent SABME) should initiate the Correlator Exchange Action.

Computation of the Correlator: The correlator is computed by concatenating the LSAP Pair Name and a time stamp or random number.

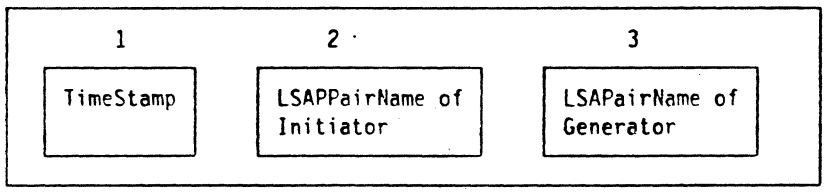


Figure 9-3. Correlator Format

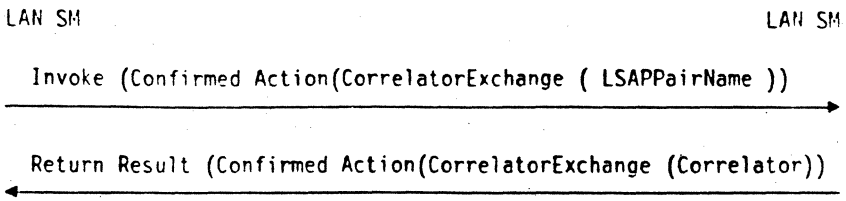
The time stamp takes the following form HHMMSS where:

HH is the decimal with range 00-23
MM is the decimal with range 00-59
SS is the decimal with range 00-59

```
X' 0A 32 1C  D4   00 11 22 33 44 55 66   D4  88 99 AA BB CC DD
   HH MM SS  SAP  |- MAC Address   -|  SAP  |- MAC Address -|
```

Figure 9-4. Example Correlator Encoding. This figure is an example encoding of the correlator, which is defined as OCTET STRING. The inputs were: TimeStamp 10:50:28, LSAP Pair Name (SAP = X'D4', MAC Address = X'00 11 22 33 44 55 66' LSAPPair Name (SAP = X'D4', MAC Address = X'88 99 AA BB CC DD').

Correlator Exchange Action Flow:



Remove Adapter

Remove Adapter is used by the LAN Manager to request that an adapter remove from the LAN. The data includes a security access field, whose use is being investigated. A station may ignore the request if it does not understand the security access field. Under normal token-ring operation, the LAN Manager will use the Force Remove MAC frame to remove an adapter from the ring. However, in non-token-ring environments, the LAN Manager will need a method to remove adapters.

CAU Wrap Action

The CAU Wrap Action is used by the LAN Manager to request that a CAU perform the wrap function. The data that is sent is the type of wrap that should be performed; merge, wrapRI, wrapRO, wrapRIRO.

Soft Reset

The Soft Reset Action is used by the LAN Manager to request that an station execute a soft reset. Future object classes may use this action also.

Remote Program Update (RPU) Enable

The Remote Program Update (RPU) Enable is used by the LAN Manager to request that a station enable the Remote Program Update Process.

Multiple Register Request

The Multiple Register Request allows a Managing Process to register with multiple functions at a station with a single Action. This is especially useful in situation where the a station performs many different functions. The data includes the following information:

- SET of
  - group function title
  - qualifier
  - distinguishing attribute
  - security access field
  - Confirmed Event Retry Timeout
  - Confirmed Event Retry Number
  - user data
- primary name
- MAC address through which this entity can be reached
- managing process title
- managing process address
- managing process level
- lost managing process retries

- managing process SAP

The response to the Multiple Register Request Action includes the following:

- SET OF
  - group function title and qualifier
  - distinguishing attribute
  - return code
  - security access field
  - character set
  - user data
- primary name
- MAC Address
- arch release level

Multiple Deregister Request

The Multiple Deregister Request Action causes the LAN Station Manager to deregister a managing process from a group of functions. The managing process is removed from the registered list for each group function, qualifier and distinguishing attribute in the Multiple register frame. The following data is in the Multiple Deregister Action:

- SET OF
  - group function title
  - qualifier
  - distinguishing attribute
  - user data
- primary name
- mac address
- managing Process Name
- managing Process Address

Chapter 10. CMIP Get

This chapter describes CMIP protocol Confirmed GET. Confirmed Get is used when the management data is required of a management entity. The management data is defined as attributes of a managed object. The values of these attributes which are defined to have read-access can be requested via the Confirmed GET by the attribute ids (object identifiers) assigned in the managed object definition. The values of the attributes are returned in the response to the Confirmed Get (See Figure 10-1.)

If a LAN Station Manager receives a GET from a station that is not a managing process with which it is registered, the LAN Station Manager will perform the GET and send a General Report (Nonregistered GET) to the managing processes with which it is registered.

If no attribute ID list is present in the CMIP frame, all attributes for that managed object instance are returned.

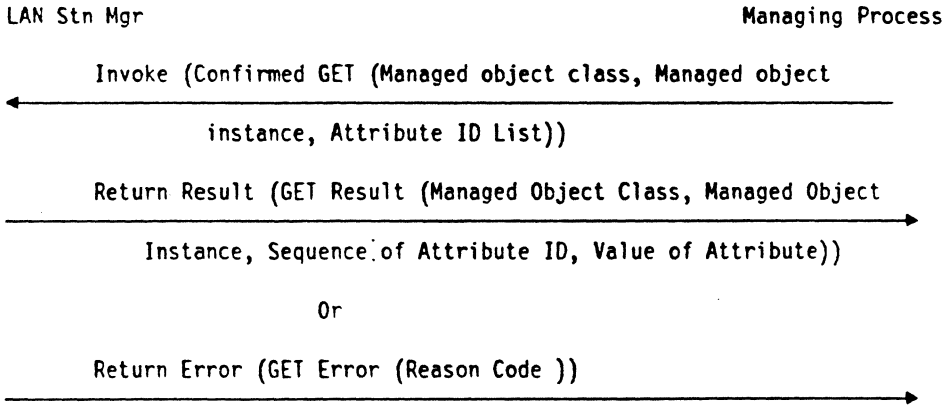


Figure 10-1. Management Flows for Confirmed Get

10.1 General Format of the Confirmed Get

CMIP Service	CMIP Parameters	Mgt Parameters
Confirmed GET	Object ID	Layer Object Class and Instance
	Attribute ID List	Attribute Object identifier
Get Result	Object ID	Layer Object Class and Instance
	Attribute List	Attribute Object Identifiers and Values

Figure 10-2. Confirmed Get and its associated parameters

There are two main fields for the Confirmed GET : the Object Id and the attribute ID list. The Get Result has two fields also: the object ID and the attribute list. These fields are discussed in this section.

Note: The objects and attributes are defined in OSI Management template format with accompanying ASN.1 definitions in 21.1, "Managed Object Templates" on page 21-1. The RO Invoke and the CMIP MPDU for the GET is defined in Chapter 16, "Protocol Data Unit ASN.1" on page 16-1.

10.1.1 Object ID

This field identifies the managed object for which the Confirmed GET is destined. It is made up of two components : Managed Object Class and Managed Object Instance.

10.1.2 Attribute ID List

The attribute ID list is a list of attribute ids for which the values are requested in the Confirmed Get. No attribute ID list in the CMIP GET signifies a request for all the attributes in the managed object instance.

10.1.3 Attribute List

The attribute list is a list of attribute ids and the values of those attributes requested in the Confirmed Get.

10.2 Confirmed GET State Diagram

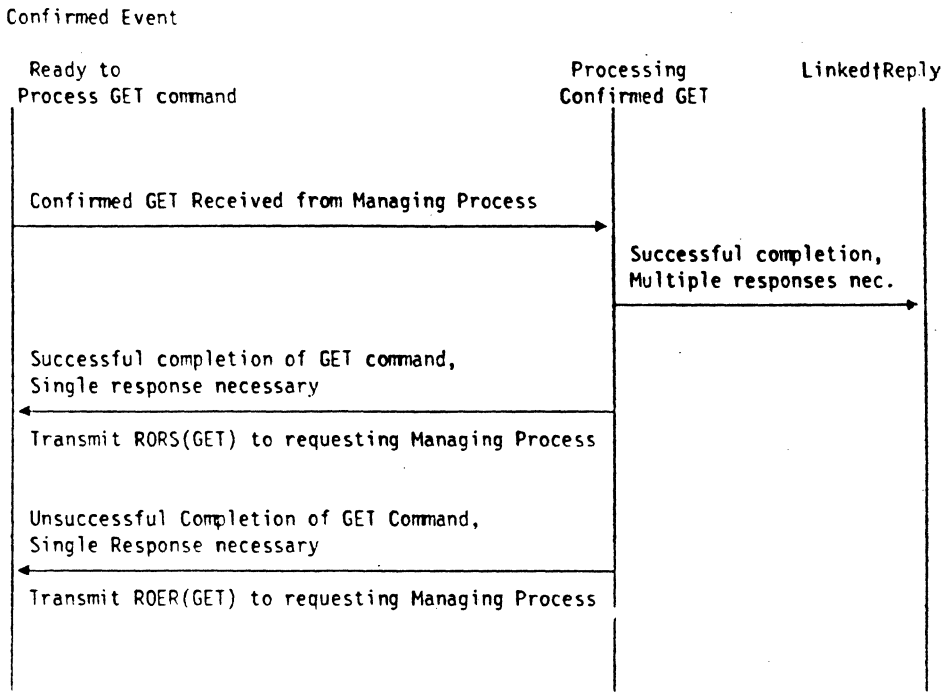


Figure 10-3. Confirmed GET State Diagram

Chapter 11. CMIP Set

This chapter describes CMIP protocol verb SET. SET is used when management in one node wants to set some management data of another node. The management data is defined as attributes of a managed object. The values of these attributes which are defined to have write-access can be set via the SET by the attribute ids (object identifiers) assigned in the managed object definition. The values of the attributes are returned in the response to the Confirmed Set (See Figure 11-1.)

The criteria for accepting a SET command from a managing process may vary by LME. Some LMEs may limit SET commands from registered managing processes; others may require password verification. An LME containing an instance of Token Ring Layer 1 object must allow an access unit to set the attributes pertaining that access unit.

In the cases where a SET is preformed, a General Report (Set Occurred) will be sent to all managing processes, except the one requesting the SET, indicating the change that was made.

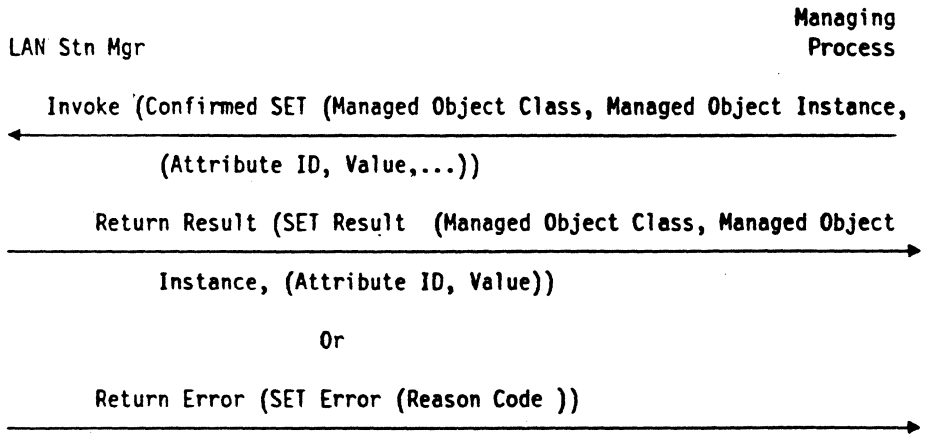


Figure 11-1. Management Flows for Confirmed Set

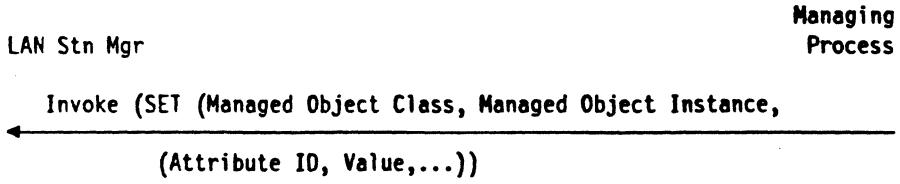


Figure 11-2. Management Flows for UnConfirmed Set

11.1 General Format of the SET

CMIP Service	CMIP Parameters	Mgt Parameters
SET	Object ID	Layer Object Class and Instance
	Modify Attribute List	Attribute id/ value pair, modify operator
Set Result	Object ID	Layer Object Class and Instance
	Attribute List	Attribute ID / value pair

Figure 11-3. Set and its associated parameters

There are two main fields for the SET and SET Result: the Object Id and the modify attribute list. These fields are discussed in this section

Note: The objects and attributes are defined in OSI Management template format with accompanying ASN.1 definitions in 21.1, "Managed Object Templates" on page 21-1. The RO Invoke and the CMIP MPDU for the Set is defined in Chapter 16, "Protocol Data Unit ASN.1" on page 16-1.

11.1.1 Object ID

This field identifies the managed object for which the Confirmed SET is destined. It made up of two components; Managed Object Class and Managed Object Instance.

11.1.2 Attribute List

The attribute list is a list of attribute ids and the values of those attributes pairs.

Chapter 12. CMIP Create

This chapter describes the LAN Management functions that use the system management CMIP protocol Confirmed Create. Figure 12-1 shows the generic flow of the Confirmed Create. CREATE flows between a Managing Process and LAN Station Manager. Create commands should be rejected with a CreateError of access denied if they do not come from a registered managing process.

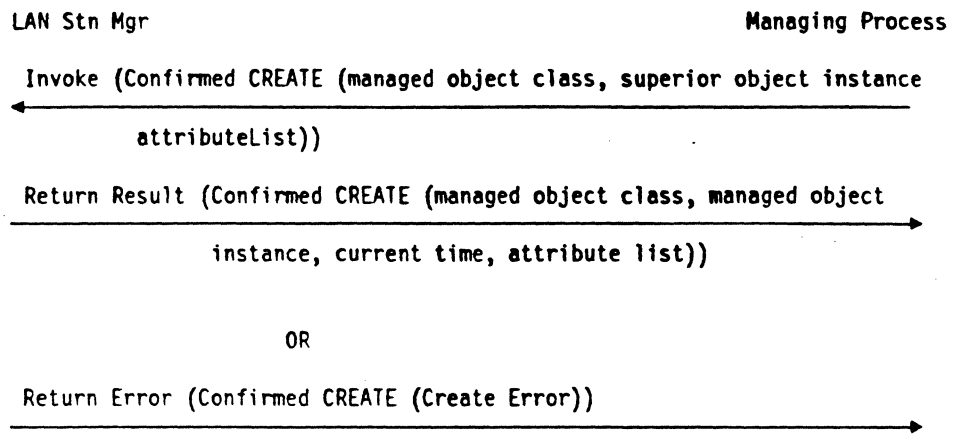


Figure 12-1. Management Flow for Creates

12.1 General Format of CREATE

CMIP Service	CMIP Parameters	Mgt Parameters
Confirmed Create	Managed Object Class	Class of new managed obj instance which will be created.
	Superior object instance	The object instance under which the new will be created.
	Reference Instance	An existing instance from which default attribute values are obtained
	AttributeList	set of attribute ids and values which should be assigned to the new managed object instance.

Figure 12-2. Confirmed Create and its associated parameters

There are three main fields for the Confirmed Create: the managed object class of the managed object instance to be created, the superior object instance under

which the object instance is to be created, and an attribute list which contains a set of attribute identifiers and values that are assigned to the new managed object instance.

Note: The objects and attributes are defined in OSI Management template format with accompanying ASN.1 definitions in 21.1, "Managed Object Templates" on page 21-1. The RO Invoke and the CMIP MPDU for the Create is defined in Chapter 16, "Protocol Data Unit ASN.1" on page 16-1.

Chapter 13. CMIP Delete

This chapter describes the LAN Management functions that use the system management CMIP protocol verb Confirmed Delete. Figure 13-1 shows the generic flow of the Confirmed Action. The Deletes will flow between a Managing Process and LAN Station Manager. Currently the only object instance that are deleted via this protocol verb is the Threshold Control Managed Object. Delete commands should be rejected if they do not come from a registered managing process.

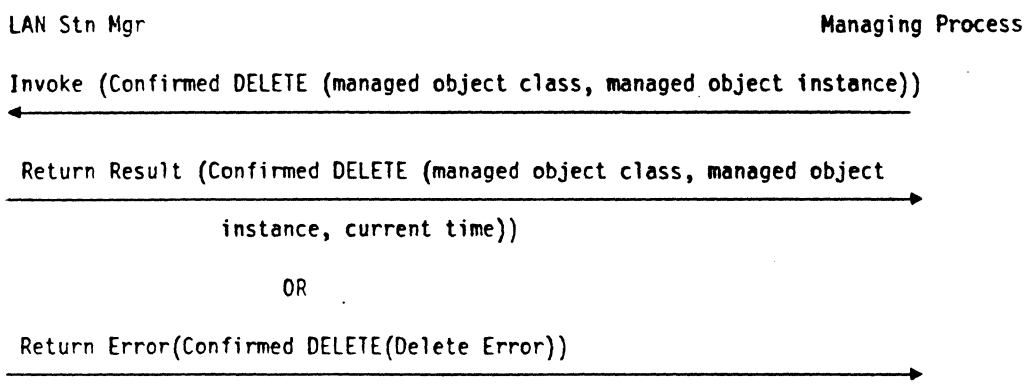


Figure 13-1. Management Flow for Delete

13.1 General Format of DELETE

CMIP Service	CMIP Parameters	Mgt Parameters
Confirmed Delete	Managed Object Class	Class of managed object class to be deleted.
	Managed Object Instance	The object instance to be deleted.

Figure 13-2. Confirmed Delete and its associated parameters

There are two fields for the Confirmed Delete: the managed object class of the managed object instance to be deleted and the managed object instance to be deleted.

Chapter 14. CMIP Linked Reply

The Linked Reply can be used to segment the response to a confirmed GET or SET.

Upon receipt of a confirmed GET, SET, ACTION, or DELETE, LAN Station Manager will determine if a response can be sent in a single PDU. If not, the linked reply operation will be used to respond to the confirmed operation.

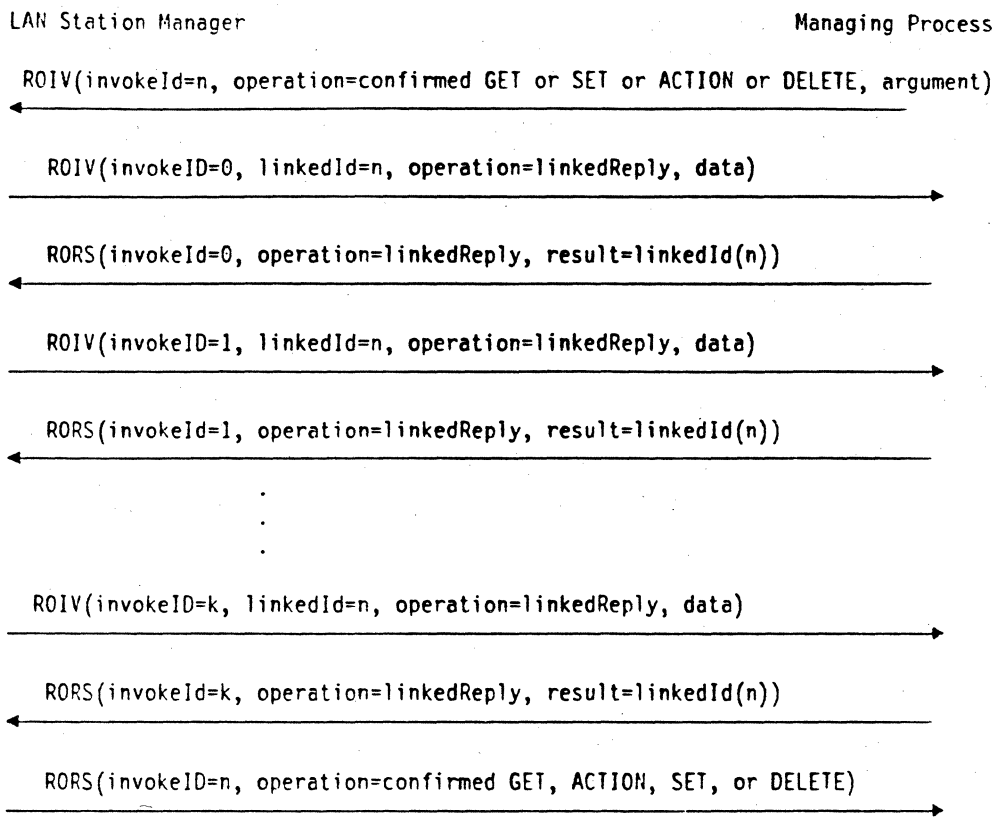
If the Station Manager decides that *n* responses are necessary, those *n* responses are sent in RO Invoke using the linked reply operation. Finally, the RORS PDU is sent via RO Result with operation confirmed GET, SET, ACTION or DELETE. The responses are segmented along attribute boundary. That is, the data in an RO Invoke using the linked reply must contain an attribute list (which consists of a set of attribute ID and attribute value).

No operation data is sent in this RO Result.

The RO Invoke PDU for the linked reply operation has an additional field after the invoke ID called the linked ID. The value of the linked ID is the invoke ID of the requesting confirmed GET or SET. The invoke ID of the linked replies should be assigned sequentially, beginning with zero, to allow for the managing process to respond to each ROIV.

Each ROIV PDU with operation of Linked Reply is sent by the LAN station manager and a response is awaited before the next ROIV PDU is transmitted. The ROIV is resent a number of times if no response is received within a retry time. If still no response is received, transmission of this response is aborted.

Once all the ROIV PDUs with operation of Linked Reply are sent, a final RO Result is sent with the invokeId equal to the invoke ID of the requesting confirmed operation. This RO Result signals that the linked reply is complete.



Note: In the case where scoping is used, ACTIONs and DELETEs can be sent as linked replies.

14.1 Linked Replay State Diagram

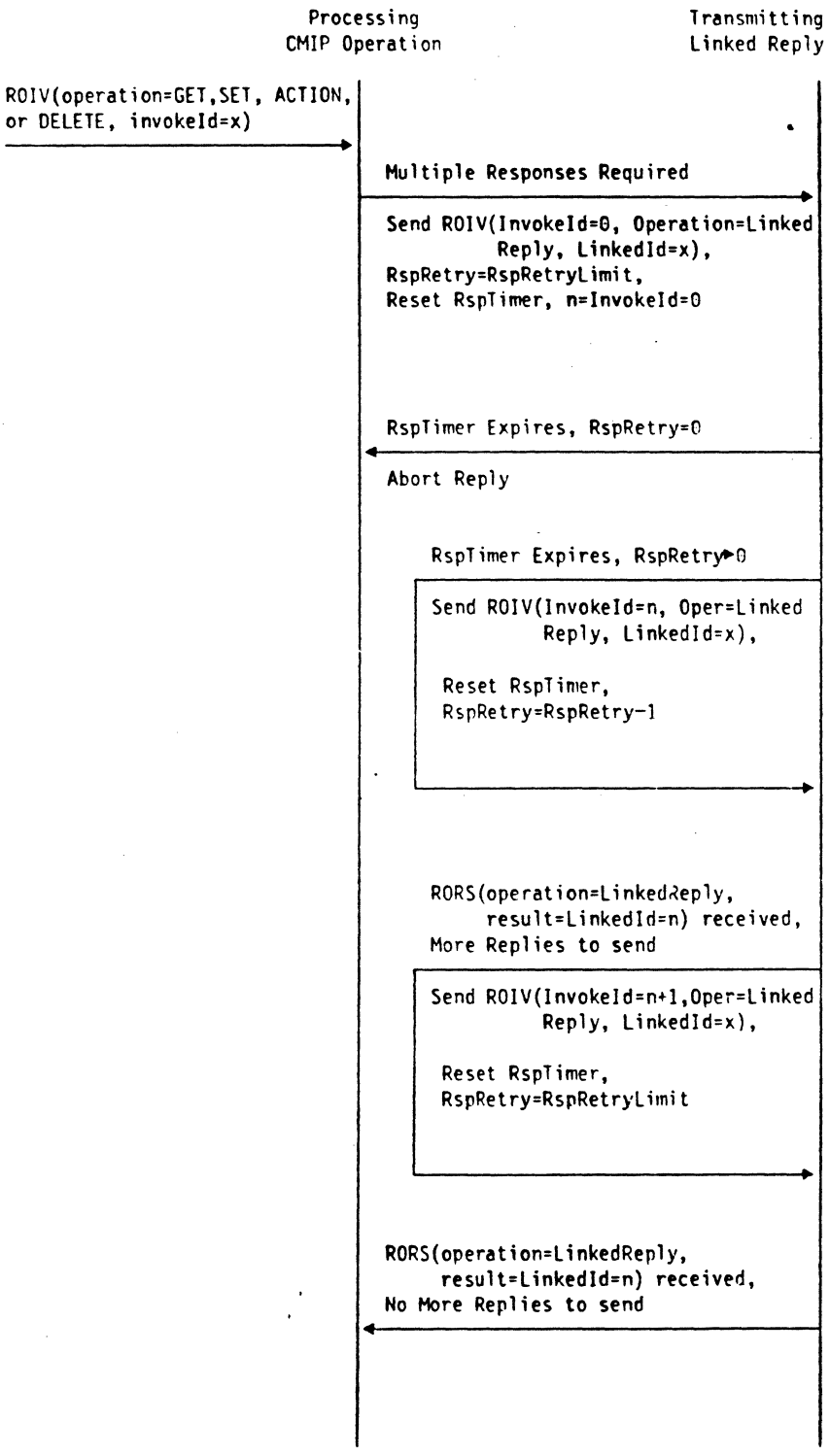


Figure 14-1. Linked Reply State Diagram

Chapter 15. CMIP Cancel Get

This chapter describes CMIP protocol CancelGet. Cancel Get is used by a managing process to request a managed entity to cancel an outstanding CMIP GET request. On receipt of a CancelGet request, the managed entity reports acceptance or rejection of the Cancel Get command. If the invokeId parameter in the Cancel Get is not the invokeId of a GET request that the LAN Station Manager is currently processing, an ROER is returned. Otherwise, the identified GET operation is cancelled by sending a ROER (error-value of operationCancelled) and an RORS for the Cancel-Get operation.

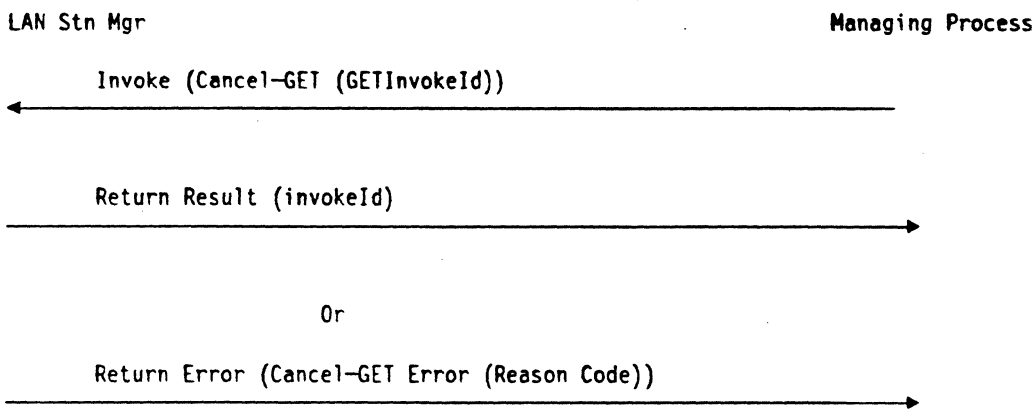


Figure 15-1. Management Flows for Cancel Get

15.1 General Format of the Cancel Get

CMIP Service	CMIP Parameters	Mgt Parameters
Cancel-GET GET	GetInvokeId	invoke ID of GET to be cancelled
Cancel-GET Result	none	

Figure 15-2. Cancel Get and its associated parameters

Cancel-GET has only one parameter, the invoke ID for the GET command that is to be cancelled. The Cancel-GET result has no parameters. It carries only the RORS invoke Id.

Note: The objects and attributes are defined in OSI Management template format with accompanying ASN.1 definitions in 21.1, "Managed Object Templates" on page 21-1. The RO Invoke and the CMIP MPDU for the Cancel Get is defined in Chapter 16, "Protocol Data Unit ASN.1" on page 16-1.

Chapter 16. Protocol Data Unit ASN.1

The following module provides for definitions that are commonly used in other chapters of this document. In some cases, these definitions replicate existing ones in standards documents. If a discrepancy occurs, the standard document, if an ISO International Standard or CCITT Recommendation, takes precedence.

16.1 REMOTE-OPERATIONS

```
REMOTE-OPERATIONS
 DEFINITIONS ::= BEGIN

 -- Remote  Operations Management protocol data units
 -- which are defined in CCITT X.229 and ISO 9072/2. The type names
 -- may differ, but the base types and encoding are the same.


 ROIVapdu ::= [1] IMPLICIT SEQUENCE{  -- Invoke PDU
          invokeId InvokeIdType,
          linkedID [0] IMPLICIT InvokeIdType OPTIONAL,
          operation-value Operation,
          argument ANY DEFINED BY operation-value}

 RORSapdu ::= [2] IMPLICIT SEQUENCE{  -- Result PDU
          invokeId InvokeIdType,
          resultOption SEQUENCE{ operation-value Operation,
          result ANY DEFINED BY operation-value} OPTIONAL}
 ROERapdu ::= [3] IMPLICIT SEQUENCE{  -- Error PDU
          invokeID InvokeIdType,
          error-value Error,
          parameter ANY DEFINED BY error-value OPTIONAL}

 RORJapdu ::= [4] IMPLICIT SEQUENCE{  -- Reject PDU
          invokeID CHOICE{InvokeIdType, NULL},
          problem CHOICE{
            [0] IMPLICIT GeneralProblem,
            [1] IMPLICIT InvokeProblem,
            [2] IMPLICIT ReturnResultProblem,
            [3] IMPLICIT ReturnErrorProblem}}

 InvokeIdType ::= INTEGER
   -- This is generated by the sender and is used to correlate the
   -- request with the corresponding response (if any).

 Operation ::=  INTEGER
   -- This field identifies operations such as Event Report, Get,
   -- Set, Action, etc. ROSE allows this data type to expand to
   -- INTEGER. However, only the INTEGER form is used in
   -- this architecture. Values for the Operation field are assigned
   -- by ISO/CCITT or by other registration authorities.

 Error ::=  INTEGER
```

```
GeneralProblem    ::= INTEGER{       -- ROSE-provider detected
                  unrecognizedAPDU(0),
                  mistypedAPDU(1),
                  badlyStructuredAPDU(2)}

InvokeProblem     ::= INTEGER{       -- ROSE-user detected
                  duplicateInvokation(0),
                  unrecognisedOperation(1),
                  mistypedArgument(2),
                  resourceLimitation(3),
                  initiatorReleasing(4),
                  unrecognizedLinkedId(5),
                  linkedResponseUnexpected(6),
                  unexpectedChildOperation(7)}

ReturnResultProblem ::= INTEGER{     -- ROSE-user detected
                  unrecognisedInvocation(0),
                  resultResponseUnexpected(1),
                  mistypedResponse(2)}

ReturnErrorProblem ::= INTEGER{      -- ROSE-user detected
                  unrecognisedInvocation(0),
                  errorResponseUnexpected(1),
                  unrecognisedError(2),
                  unexpectedError(3),
                  mistypedParameter(4)}

END
```

16.2 CMIP

```
CMIP
 DEFINITIONS ::= BEGIN
 IMPORTS Operation FROM REMOTE-OPERATIONS
 EXPORTS ibmISDN, EventReportArgument, ObjectClass, ObjectInstance,
      EventTypeId, AVA

 -- The following OBJECT IDENTIFIER value forms the stem for other OBJECT
 -- IDENTIFIER values used throughout this architecture.
 -- NOTE NOTE NOTE: This stem has not been allocated by an authorized
 -- registration authority.  When a properly registered value becomes
 -- available, it will replace all this definition, implying
 -- corresponding changes to other OBJECT IDENTIFIERs which make
 -- reference to it.

 ibmISDN OBJECT IDENTIFIER ::= {iso(1) registration-authority(1)
                   ansi(1) ibm(99)}

 -- CMIP protocol data units are taken from ISO 9596-2.  Data type
 -- names may differ, but the base types and resulting encodings
 -- should correspond to those defined in ISO 9596-2.

 -- The following defines the CMIP unconfirmed Event Report.
 -- It uses the EventReportArgument data type for the argument element
```

```
-- of the ROIVapdu

eventReportOp Operation ::= 0 -- mapped to operation-value

-- The following defines the CMIP confirmed Event Report and its
-- result. It uses the EventReportArgument data type for the
-- argument element of ROIVapdu. The EventReportResult is optional;
-- if used, it flows in the result element of the RORSapdu.

eventReportConfirmedOp Operation ::= 1 -- mapped to
 -- operation-value of ROIVapdu

-- See LAN-Notifies for the details of Event

EventReportArgument ::= -- mapped to argument of ROIVapdu
   SEQUENCE{
    managedObjectClass     ObjectClass,
    managedObjectInstance  ObjectInstance,
    eventTime              [5] IMPLICIT GeneralizedTime OPTIONAL,
    eventType              EventTypeId,
    eventData              [8] ANY DEFINED BY eventType OPTIONAL
     -- NOTE: for specific event types, eventData may not
     -- be optional
   }

EventReportResult ::= -- mapped to result of RORSapdu
   SEQUENCE{
    managedObjectClass ObjectClass OPTIONAL,
    managedObjectInstance  ObjectInstance OPTIONAL,
    currentTime            [5] IMPLICIT GeneralizedTime OPTIONAL,
    eventReply             SEQUENCE {
      eventType            EventTypeId,
      eventResult          [8] ANY DEFINED BY eventType OPTIONAL} OPTIONAL }

EventError ::= CHOICE{
            NoSuchObjectClass,   -- noSuchObjectClass,
            NoSuchObjectInstance, -- noSuchObjectInstance,
            NoSuchEventType,     -- noSuchEventType,
            NoSuchArgument,      -- noSuchArgument,
            InvalidArgumentValue, -- invalidArgumentValue,
            ProcessingFailure}   -- processingFailure

-- The following defines the CMIP confirmed Get.
-- It uses the GetArgument data type for the argument element
-- of the ROIVapdu. The result is returned by GetResult, which
-- is conveyed as the result element of RORSapdu. If an error
-- occurs, GetError is returned in parameter element of ROERapdu.
-- The corresponding error-value element is specified along with
-- the parameter.

getOp Operation ::= 3

GetArgument ::= SEQUENCE {
    managedObjectClass ObjectClass,
    managedObjectInstance ObjectInstance,
    accessControl [5] AccessControl OPTIONAL,
```

```
    synchronization [6] IMPLICIT CMISSynch DEFAULT bestEffort,
    scope    [7] Scope DEFAULT baseObject,
    filter   CMSFilter DEFAULT and { },
    attributeIdList [12] IMPLICIT SET OF AttributeId OPTIONAL}

GetResult ::= SEQUENCE {
    managedObjectClass ObjectClass OPTIONAL,
    managedObjectInstance ObjectInstance OPTIONAL,
    currentTime [5] IMPLICIT GeneralizedTime OPTIONAL,
    attributeList [6] IMPLICIT SET OF Attribute OPTIONAL }


GetError ::= CHOICE{
            ClassInstanceConflict -- classInstanceConflict
            ComplexityLimitation  -- complexityLimitation
            InvalidFilter       -- invalidFilter
            InvalidScope        -- invalidScope
            NoSuchObjectClass,   -- noSuchObjectClass,
            NoSuchObjectInstance, -- noSuchObjectInstance,
            AccessDenied,       -- accessDenied,
            GetListError,       -- getListError,
            ProcessingFailure    -- processingFailure
            SyncNotSupported}    -- syncNotSupported (not used)

-- The following defines the CMIP unconfirmed Set.
-- It uses the SetArgument data type for the argument element
-- of the ROIVapdu.

setOp Operation ::= 4

-- The following defines the CMIP confirmed Set.
-- of the ROIVapdu.  The result is returned by SetResult, which
-- is conveyed as the result element of RORSapdu.  If an error
-- occurs, SetError is returned in parameter element of ROERapdu.
-- The corresponding error-value element is specified along with
-- the parameter.
setConfirmedOp Operation ::= 5

SetArgument ::= SEQUENCE {
    managedObjectClass ObjectClass,
    managedObjectInstance ObjectInstance,
    accessControl [5] AccessControl OPTIONAL,
    synchronization [6] IMPLICIT CMISSynch DEFAULT bestEffort,
    scope [7] Scope DEFAULT baseObject,
    filter CMISFilter DEFAULT and { }
    modifyattributeList [12] IMPLICIT SET OF SEQUENCE {
     attributeId AttributeId,
     attributeValue [2] ANY DEFINED BY AttributeId OPTIONAL,
     modifyOperator [3] ModifyOperator DEFAULT replace}
    }

ModifyOperator ::= ENUMERATED{
            replace(0),
            addValues(1),
            removeValues(2),
            setToDefault(3)}
```

```
SetResult ::= SEQUENCE {
    managedObjectClass ObjectClass OPTIONAL,
    managedObjectInstance ObjectInstance OPTIONAL,
    currentTime [5] IMPLICIT GeneralizedTime OPTIONAL,
    attributeList [6] IMPLICIT SET OF Attribute OPTIONAL}

SetError ::= CHOICE{
            ClassInstanceConflict -- classInstanceConflict (not used),
            ComplexityLimitation  -- complexityLimitation (not used),
            InvalidFilter       -- invalidFilter (not used),
            INvalidScope        -- invalidScope (not used),
            NoSuchObjectClass,   -- noSuchObjectClass,
            NoSuchObjectInstance, -- noSuchObjectInstance,
            AccessDenied,        -- accessDenied,
            SetListError,       -- setListError,
            ProcessingFailure    -- processingFailure
            SyncNotSupported     -- syncNotSupported (not used),
            InvalidOperator     -
            InvalidOperation}   -


-- The following defines the CMIP unconfirmed Action.  It is
-- conveyed in the operation-value element of ROIVapdu.

actionOp Operation ::= 6

-- The following defines the CMIP confirmed Action.  It is
-- conveyed in the operation-value element of ROIVapdu.

actionConfirmedOp Operation ::= 7

-- The following is the action argument.  It is conveyed as the
-- argument element of ROIVapdu.

ActionArgument ::= SEQUENCE {
    managedObjectClass ObjectClass,
    managedObjectInstance ObjectInstance,
    accessControl [5] AccessControl OPTIONAL,
    syncronization [6] IMPLICIT CMISSynch DEFAULT bestEffort,
    scope [7] Scope DEFAULT baseObject,
    filter CMSFilter DEFAULT and {,
    actionData [12] IMPLICIT ActionData}

ActionData ::= SEQUENCE {
    actionType ActionTypeId,  -- this is defined by the managed
                    -- object class in the ACTION
                    -- template
    actionDataArg [4] ANY DEFINED BY actionType OPTIONAL -- this
                    -- is defined by the ACTION template
                    -- for the object class
                    }

-- The result (which is optional) for an action is conveyed by
```

```
-- the following data type, it maps to the result element of
-- RORSapdu

ActionResult ::= SEQUENCE {
    managedObjectClass ObjectClass OPTIONAL,
    managedObjectInstance   ObjectInstance OPTIONAL,
    currentTime [5] IMPLICIT GeneralizedTime OPTIONAL,
    actionResultData [6] IMPLICIT ActionResultData OPTIONAL}

ActionResultData ::= SEQUENCE{
    actionType ActionTypeId, -- this is defined by the managed
                    -- object and the supporting defs,
                    -- specifically the ACTION templates
    actionResultArg [4] ANY DEFINED BY actionType OPTIONAL -- this
                    -- is defined by the object class for
                    -- the specific action, which appears
                    -- in an action template.
                    {

ActionTypeId ::= CHOICE{
            globalId [2] IMPLICIT OBJECT IDENTIFIER,
            localForm  [3] IMPLICIT INTEGER (not used)}

ActionError ::= CHOICE{
            ClassInstanceConflict -- classInstanceConflict (not used),
            ComplexityLimitation  -- complexityLimitation (not used),
            InvalidScope        -- invalidScope (not used),
            InvalidFilter       -- invalidFilter (not used),
            NoSuchObjectClass,   -- noSuchObjectClass,
            NoSuchObjectInstance, -- noSuchObjectInstance,
            AccessDenied,        -- accessDenied,
            NoSuchAction,        -- noSuchAction
            NoSuchArgument,      -- noSuchArgument,
            InvalidArgumentValue, -- invalidArgumentValue,
            ProcessingFailure    -- processingFailure,
            SyncNotSupported}    -- syncNotSupported (not used)



-- The following defines the CMIP confirmed Create,
-- of the ROIVapdu.  The result is returned by CreateResult, which
-- is conveyed as the result element of RORSapdu.
createOp Operation ::= 8

CreateArgument ::= SEQUENCE {
   ManagedObjectClass ObjectClass,
              CHOICE {
                managedObjectInstance ObjectInstance,
                superiorObjectInstance [8] ObjectInstance} OPTIONAL
    accessControl [5] AccessControl OPTIONAL,
    referenceObjectInstance [6] ObjectInstance OPTIONAL,
    attributeList  [7] IMPLICIT SET OF Attribute OPTIONAL }

CreateResult ::= SEQUENCE {
    managedObjectClass ObjectClass OPTIONAL,
```

```
    managedObjectInstance ObjectInstance OPTIONAL,
     --Must be returned if omitted form Create Argument
    currentTime [5] IMPLICIT GeneralizedTime OPTIONAL,
    attributeList [6] IMPLICIT SET OF Attribute OPTIONAL}

CreateError ::= CHOICE{
            ClassInstanceConflict -- classInstanceConflict (not used),
            NoSuchObjectClass,   -- noSuchObjectClass,
            DuplicateManagedObjectInstance, -- duplicateManagedObjectInstance,
            NoSuchReferenceObject, -- noSuchReferenceObject,
            AccessDenied,        -- accessDenied,
            NoSuchAttribute      -- noSuchAttribute,
            NoSuchObjectInstance -- noSuchObjectInstance (not used),
            InvalidAttributeValue -- invalidAttributeValue,
            InvalidObjectInstance -- invalidObjectInstance (not used),
            MissingAttributeValue -- missingAttributeValue (not used),
            ProcessingFailure}   -- processingFailure

-- The following defines the CMIP confirmed Delete,
-- of the ROIVapdu. The result is returned by DeleteResult, which
-- is conveyed as the result element of RORSapdu.

deleteOp Operation ::= 9

DeleteArgument ::= SEQUENCE {
    managedObjectClass ObjectClass,
    managedObjectInstance ObjectInstance,
    accessControl [5] AccessControl OPTIONAL,
    synchronization  [6] IMPLICIT CMISSynch DEFAULT bestEffort,
    scope [7] Scope DEFAULT baseObject,
    CMSFilter DEFAULT and {},          }



DeleteResult ::= SEQUENCE {
   managedObjectClass ObjectClass OPTIONAL,
   managedObjectInstance ObjectInstance OPTIONAL,
   currentTime [5] IMPLICIT GeneralizedTime OPTIONAL }

DeleteError ::= CHOICE{
            ClassInstanceConflict -- classInstanceConflict (not used),
            ComplexityLimitation  -- complexityLimitation (not used),
            InvalidFilter        -- invalidFilter (not used),
            InvalidScope         -- invalidScope (not used),
            NoSuchObjectClass,   -- noSuchObjectClass,
            NoSuchObjectInstance, -- noSuchObjectInstance,
            AccessDenied,        -- accessDenied,
            ProcessingFailure    -- processingFailure,
            SyncNotSupported}    -- syncNotSupported (not used)

LinkedReply Operation ::= 2

LinkedReplyArgument ::= CHOICE {
   getResult       [0] IMPLICIT GetResult,
   getError        [1] IMPLICIT GetListError,
   setResult       [2] IMPLICIT SetResult,
```

```
   setError        [3] IMPLICIT SetListError,
   actionResult    [4] IMPLICIT ActionResult,
   processingFailure [5] IMPLICIT ProcessingFailure,
   deleteResult    [6] IMPLICIT DeleteResult,
   actionError     [7] IMPLICIT ActionError,
   deleteError     [8] IMPLICIT DeleteError }

LinkedReplyResult ::= InvokeIdType  -- the value is the linkedId

cancelGet Operation ::= 10

CancelGetArgument ::= GetInvokeId

GetInvokeId ::= InvokeIdType

CancelGetError ::= CHOICE{
          MistypedOperation,
          NoSuchInvokeId }


-- The following are supporting definitions

ObjectClass ::= CHOICE{
          globalId [0] IMPLICIT OBJECT IDENTIFIER
          -- nonSpecificForm [1] IMPLICIT INTEGER (not used)
          }

ObjectInstance ::= CHOICE{
            distinguishedName [2] IMPLICIT DistinguishedName
            -- nonSpecificForm not used
            localDistinguishedName [4] IMPLICIT SEQUENCE OF RDN
            }

DistinguishedName ::= SEQUENCE OF RDN -- RelativeDistinguishedName

RDN  ::=  SET OF AVA -- AttributeValueAssertion

AVA  ::=  SEQUENCE { attributeId AttributeId,
            attributeValue ANY DEFINED BY attributeId }

AttributeId  ::=  CHOICE {globalForm[0]  IMPLICIT OBJECT IDENTIFIER,
           localForm [1] IMPLICIT INTEGER (not used) }

Attribute ::= SEQUENCE {
          attributeId AttributeId,
          attributeValue ANY DEFINED BY attributeId}

EventTypeId ::= CHOICE{
          globalId [6] IMPLICIT OBJECT IDENTIFER
          -- localId [7] IMPLICIT INTEGER  (not used)
          }

AccessControl ::= EXTERNAL

Scope ::= CHOICE {INTEGER {baseObject(0),
                    firstLevelOnly(1),
```

```
                    wholeSubtree (2), },
                individualLevels [1] IMPLICT INTEGER(0..MAX),
                baseToNthLevel   [2] IMPLICT INTEGER(0..MAX).}

CMISSync ::= ENUMERATED {bestEffort(0),
                         atomic(1) }

CMISFilter ::= CHOICE
  { item [8] FilterItem,
      and  [9] IMPLICIT SET OF CMISFilter,
      or   [10] IMPLICIT SET OF CMISFilter,
      not  [10] CMISFilter }

FilterItem ::= CHOICE
 {equality              [0] IMPLICIT Attribute,
    substrings          [1] IMPLICIT SEQUENCE OF CHOICE
     {initialString [0] IMPLICIT SEQUENCE {
           attributeId AttributeID,
           string      ANY DEFINED BY attributeId },
        anyString [1] IMPLICIT SEQUENCE {
           attributeId AttributeID,
           string      ANY DEFINED BY attributeId }
        finalString [1] IMPLICIT SEQUENCE {,
          attributeId AttributeID,
          string      ANY DEFINED BY attributeId }},
    greaterOrEqual      [2] IMPLICIT Attribute,
    lessOrEqual         [3] IMPLICIT Attribute,
    present             [4] AttributeId,
    subsetOf            [5] IMPLICIT Attribute,
    supersetOf          [6] IMPLICIT Attribute
    nonNullIntersection [7] IMPLICIT Attribute }

noSuchObjectClass INTEGER ::= 0
NoSuchObjectClass ::= ObjectClass

noSuchObjectInstance INTEGER ::= 1
NoSuchObjectInstance ::= ObjectInstance

accessDenied INTEGER ::= 2

getListError INTEGER ::= 7
GetListError ::= SEQUENCE {
           ObjectClass OPTIONAL,
           ObjectInstance OPTIONAL,
           currentTime [5] IMPLICIT GeneralizedTime OPTIONAL,
           [6] IMPLICIT SET OF GetInfoStatus}

GetInfoStatus ::= CHOICE {
           attributeIdError [0] IMPLICIT AttributeIdError,
           attribute [1] IMPLICIT Attribute}

AttributeIdError ::= SEQUENCE{
             errorStatus GetErrorStatus,
             attributeId AttributeId}

GetErrorStatus ::= ENUMERATED{
```

```
                accessDenied (2),
                noSuchAttribute (5)}

setListError INTEGER ::= 8
SetListError ::= SEQUENCE {
            ObjectClass OPTIONAL,
            ObjectInstance OPTIONAL,
            currentTime [5] IMPLICIT GeneralizedTime OPTIONAL,
            [6] IMPLICIT SET OF SetInfoStatus}

SetInfoStatus ::= CHOICE {
            attributeError [0] IMPLICIT AttributeError,
            attribute [1] IMPLICIT Attribute}

AttributeError ::= SEQUENCE{
              errorStatus SetErrorStatus,
              attributeId AttributeId,
              attributeValue ANY DEFINED BY AttributeId }

SetErrorStatus ::= ENUMERATED{
            accessDenied (2),
            noSuchAttribute (5),
            invalidAttributeValue(6),
            invalidOperator(19),
            invalidOperation(20)}

noSuchAction INTEGER ::= 9
NoSuchAction ::= SEQUENCE{ ObjectClass, ActionTypeId }

processingFailure INTEGER ::= 10
ProcessingFailure ::= SEQUENCE{
              managedObjectClass ObjectClass,
              ObjectInstance OPTIONAL,
              specificErrorInfo ANY DEFINED BY
                managedObjectClass}

noSuchEventType INTEGER ::= 13
NoSuchEventType ::= SEQUENCE{ ObjectClass, EventTypeId}

noSuchArgument INTEGER ::= 14
NoSuchArgument ::= CHOICE{
        actionId [0] IMPLICIT SEQUENCE{
          ObjectClass OPTIONAL,
          ActionTypeId},
        eventId [1] IMPLICIT SEQUENCE{
          ObjectClass OPTIONAL,
          EventTypeId}}

invalidArgumentValue INTEGER ::= 15
InvalidArgumentValue ::= CHOICE{
              actionData [0] IMPLICIT ActionData,
              eventData [1] IMPLICIT SEQUENCE{
                eventTypeId EventTypeId,
                eventDataArg [8]ANY DEFINED BY eventTypeId}}

duplicateObjectInstance INTEGER ::= 16
```

```
DuplicateObjectInstance ::= OjbectInstance

noSuchReferenceObject INTEGER ::= 17
NoSuchReferenceObject ::= ObjectInstance

mistypedOperation INTEGER ::= 21

noSuchInvokeId INTEGER ::= 22
NoSuchInvokeId ::= InvokeIdType


END
```

16.3 LAN Specific ASN.1 Definitions

See Chapter 23, "ASN.1 Definitions" on page 23-1

Chapter 17. Registration

In order to allow for management of an end-station to begin, it is necessary for two-way communication between the end-station and the managing process to take place. The managing process must know that the station is attached to the LAN and available for management. The end-station must know to which, if any, managing process(es) to send events.

The registration process provides these functions. In addition, it provides for the exchange of pertinent information in both directions.

17.1 Definitions

Function Present Event - Event from a station that announces the presence of a function in that station that is in need of management.

group function title - A title that indicates the type of function the entity performs (for example, Token-Ring MAC and LLC, CAU)

qualifier - A distinguishing characteristic on group function title used to limit the scope of a management domain (for example, segment number for Token-Ring MAC).

distinguishing attribute - An attribute that distinguishes one instance of a function from another. For example, MAC address is a distinguishing attribute for the Token-Ring group function title.

primary name - A unique name for the system in which a LAN Station Manager resides. In cases where a universally-administered address exists, the primary name should be a universally-administered address.

MAC address- The MAC address through which the managed entity can be reached.

Lost Managing Process retries- The number of times an end-station should go through the sequence of finding another route to a registered managing process.

Registered List - A table of group function title, qualifier, distinguishing attribute, primary name, MAC address, and managing processes' name, address and thresholds.

Register Check - An action sent from the managing process to the stations to query the registration status of a function

17.2 Overview of Registration

When a function on a station (for example, Token-Ring MAC and LLC, CAU, Bridge) is initialized, the station manager sends a Function Present event to announce the presence of that function at the station. (If the SAP to which this event is addressed has not been discovered, the station manager should send a FIND to discover the Function Present SAP). The Function Present Event is sent to the LAN Manager functional address single route broadcast, so that all managing processes will receive it.

If the managing process is responsible for management of this function (based on function title and qualifier), it will begin the registration process by sending the station a Register Request Action. (If the managing process does not have a route to the station, it will perform a FIND first to find the route to the station.) The station will respond to the Register Request action with the appropriate data.

See Chapter 6, "Dynamic Address Resolution and Route Discovery" on page 6-1 for details on the FIND and FOUND frames.

Figure 17-1 on page 17-3 illustrates the registration process.

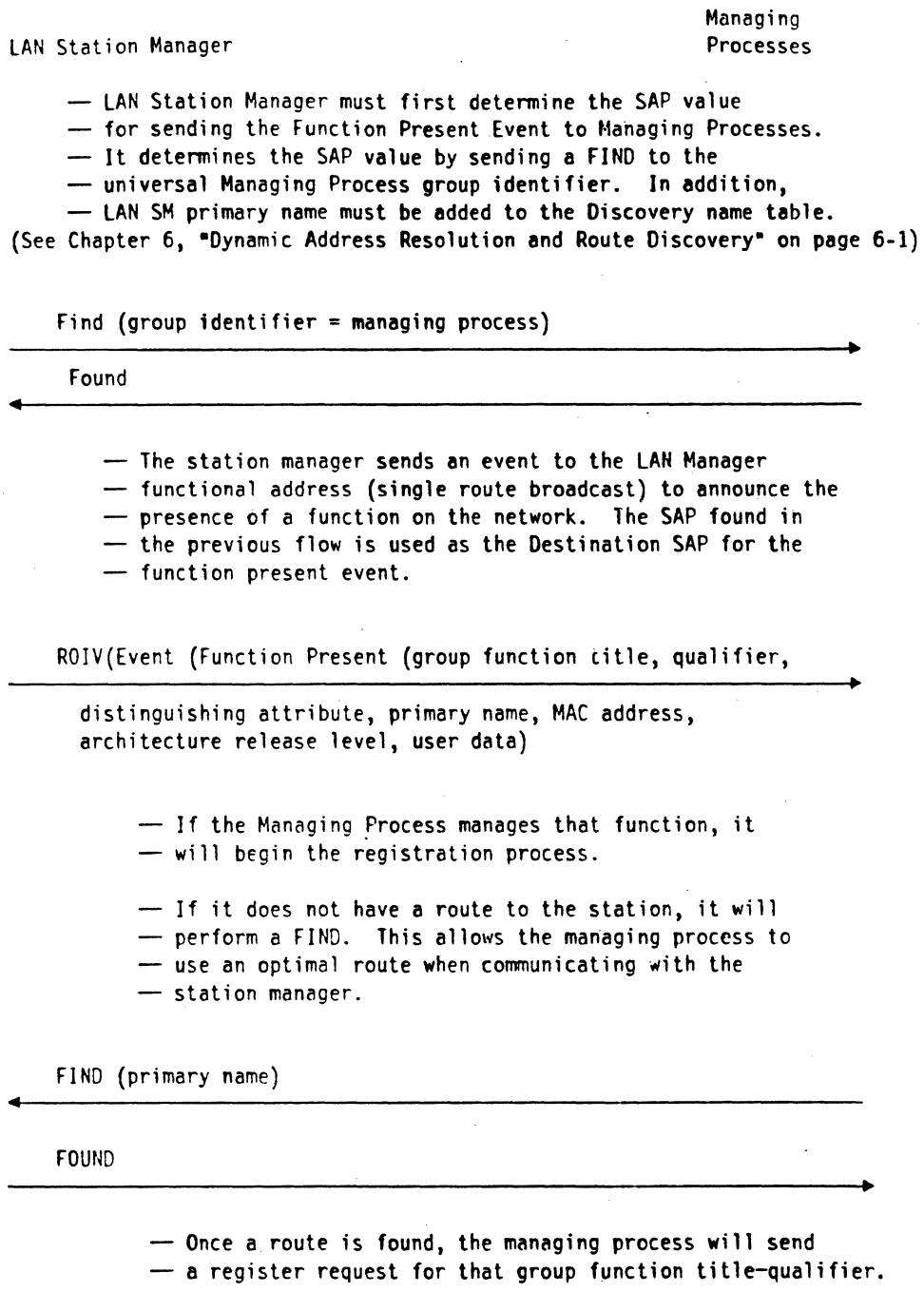


Figure 17-1 (Part 1 of 2). Registration Flow

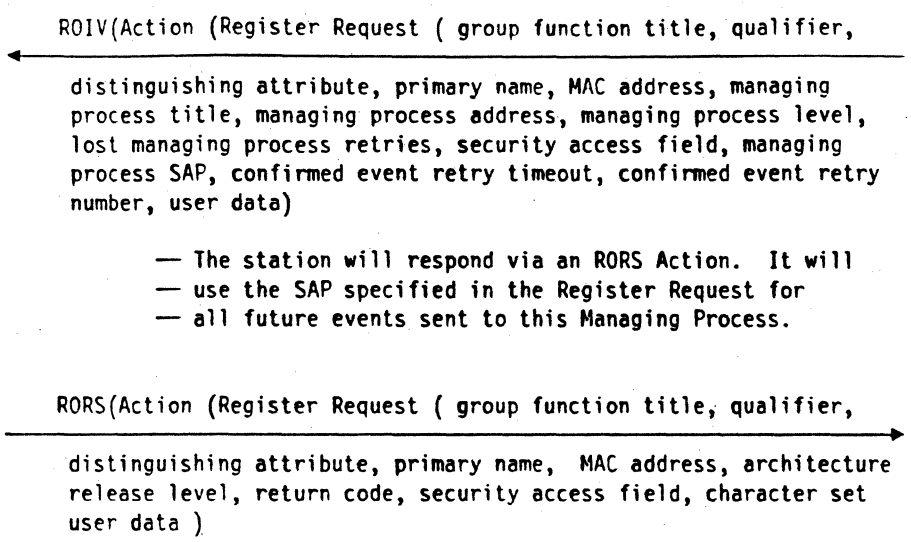


Figure 17-1 (Part 2 of 2). Registration Flow

This function is now registered with this Managing Process. It will continue to send its events to this managing process until it receives a DEREGISTER action or reaches its lost LANM Retry limit. It uses the managing process SAP specified in the Register Request frame.

Multiple Register Requests may be received for a single function. All should be accepted until there is no more space in the registered list.

The Managing Process may optionally specify the Confirmed Event Retry Timeout and Confirmed Event Retry Number on the Register Request. These numbers are used to retry a confirmed event when no confirmation is received. If no confirmation is received in the time period specified by "Confirmed Event Retry Timeout," the confirmed event is resent. This sequence is repeated until a confirmation is received or the event has been retried the number specified by "Confirmed Event Retry Number." If the Managing process does not specify these values on the Register Request, the default values should be used. The default value for the Confirmed Event Retry Timeout is the same as the Find Timer, defined in Reference Chapter 6, "Dynamic Address Resolution and Route Discovery" on page 6-1 The default value for the Confirmed Event Retry Number is the same as the Find Try Count, defined in Chapter 6, "Dynamic Address Resolution and Route Discovery" on page 6-1.

If the function being registered uses a character set other than ISO 8859-1 Latin Alphabet No. 1 for encoding graphic string types, that character set should be specified in the Register Request Response.

It is possible that the qualifier at a station manager is not set and has a null value. For example, if a token-ring station inserts into a dual ring that is wrapped and it does not get its ring number set, that station will have a ring number of zero. The ring number of zero is a null value in the token ring function. However, a managing process could know the correct qualifier and wants to register with a station

that did not get its qualifier set (null qualifier). To allow this to happen, the station manager should ignore the qualifier field in the Register Request if its qualifier is null.

Return code in the Register Request response has one of the following values:

- Positive response
- Positive response - reregistered

  Note: If a station receives a Register Request from a managing process that it already has in its registered list (same managing process title and managing process address), the registered list should be updated to contain the values of the managing process SAP and lost managing process retries contained in the most recently received Register Request.
- Group identifier not present
- No more space in Registered list
- Access denied
- Qualifier does not match.

17.3 Station Manager Registration State Diagram

The state diagram illustrates the functions of the station manager with respect to registration.

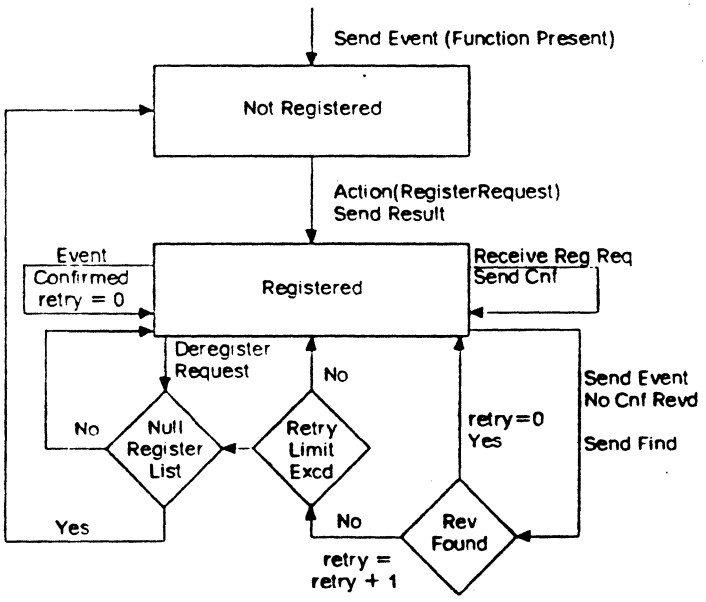


Figure 17-2. Station Manager Registration State Diagram

17.3.1 Description of States

NOT REGISTERED: While in this state, the station manager will send no events when it receives notification from the function layer management entity. Transition is made to the REGISTERED state when a Register Request is received and a response is sent.

REGISTERED: While in this state, the station manager sends events to the managing processes with which the function is registered when it receives a notification from the function's layer management entity.

17.3.2 Description of the State Diagram Transitions:

Send Event (Function Present)

A function attaches to the station manager to request management, providing group function title, qualifier and other information. Station Manager sends a Function Present event to announce the presence of the function which has just attached. Until a Register Request is received, that function is in the NOT REGISTERED State. The Event (Function Present) is sent only one time.

Action(RegisterRequest)

Upon receipt of a Register Request Action, the station manager registers that function with the managing process.

Send Result

The station manager responds to the managing process with the appropriate return code and transitions to REGISTERED State.

Receive Reg Req

Send Cnf

If station manager receives a Register Request Action for a function that already has registered, it will register that function if there is space in the register list and respond with a Register Request response to the managing process which sent the Register Request. The retry is set equal to zero.

Send Event

When a confirmed event is sent, a confirmation is expected. If no confirmation is received after the timeout period the length of the Confirmed Event Retry Timeout, the event is resent. The sequence is repeated until a confirmation is received or the Confirmed Event Retry Number is reached.

No Cnf Rcvd

Events are sent to all managing processes in the registered list for this function.

If the event confirmation is not received after the number of retries specified by the "Confirmed Event Retry Number," then:

Send Find The station manager will attempt to find a different route to the managing process by sending out a FIND to the managing process title received in the RegisterRequest frame.

Rvc FOUND?

Yes, retry=0

If a FOUND is received, the route is updated, the lost LANM retry count (rt) is set to zero, and the station manager returns to REGISTERED state.

Rcv FOUND?

No, retry = retry + 1

If a FOUND is not received (after the normal number of FIND retries), the lost LANM retry count (rt) is incremented.

Retry Limit Excd

No

If the retry limit is not exceeded, the station manager returns to REGISTERED state.

Retry Limit Excd

Yes If the lost managing process retry count is exceeded, the managing process is removed from register list and a check is performed on the register list to see if it is empty. When the managing processes is removed from the register list, all thresholds that the managing process has created are deleted.

Deregister Request

If a Deregister Request Action is received, the managing process is removed from the registered list and a check is performed on the register list to see if it is empty. When the managing processes is removed from the register list, all thresholds that the managing process has created are deleted.

Null Register List

No

If the register list is not empty (meaning the function is registered with another managing process), the station manager returns to REGISTERED state.

Null Register List

Yes

If the register list is empty, the station manager returns to NOT REGISTERED for this function.

Event Confirmed

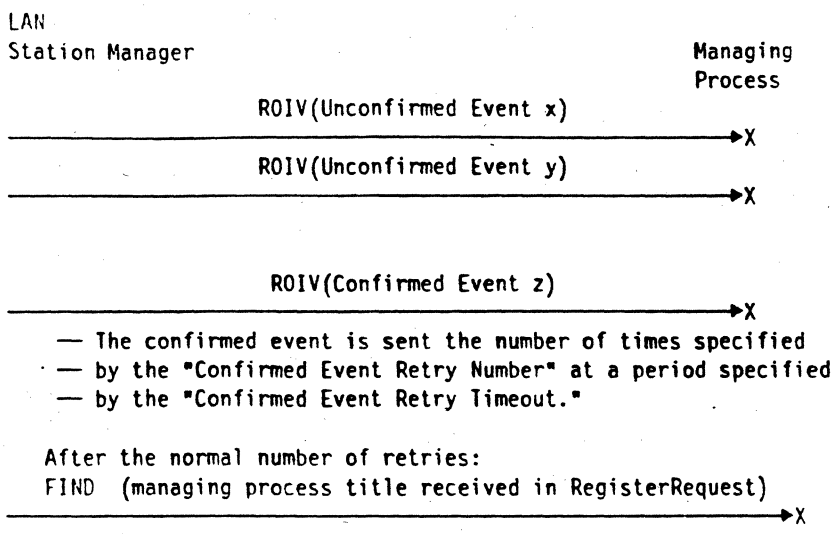
retry = 0

When the station manager receives an event to send from a function, it sends it to the managing processes with which that function is registered. If a confirmation is received, it returns to REGISTERED state. If it is an unconfirmed event, it returns to REGISTERED state.

17.3.3 Lost Managing Process Retry Loop

Figure 17-3 on page 17-8 illustrates the lost LANM retry loop in the state diagram.

While registered, stations will send events to the registered managing processes as the need arises. Occasionally the events are confirmed events. If a station does not receive a confirmation on one of its confirmed events, it will attempt to find an alternate route. If it does not find one, it will continue to send its events until the retry limit for finding a lost managing process is reached.



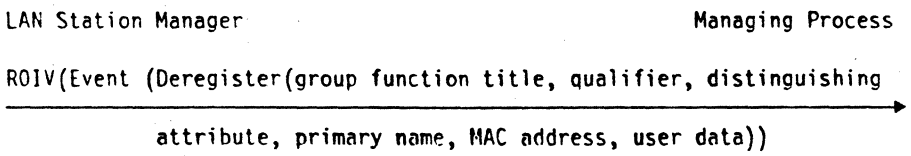
Repeat this sequence as new events are emitted until the retry limit is reached and then become unregistered.

Figure 17-3. Lost Managing Process Retry Loop

The retry loop keeps stations from sending the events endlessly when the managing process goes down. It is the managing process's responsibility to check the registration on stations after it has gone down or detects that a topology change has taken place on the network. Events that do not reached the Managing process will NOT be saved by the station manager for reporting when the managing process re-registers.

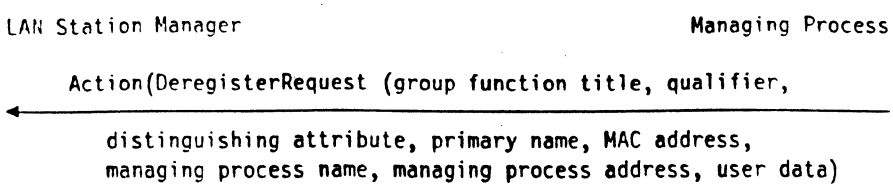
Detachment of a Function from the Station Manager.

When a function no longer requires management by way os the station manager, the station manager shall send a Deregister Event to managing processes with which that function was registered and delete group function title and all related data from its register list. This event is unconfirmed.



17.3.4 Receipt of Deregister Request Action

When a station manager receives a Deregister Request Action from a managing process, it shall delete the registration for that function and qualifier with that managing process, if such a registration exists.



17.4 Managing Process's Role in Registration

17.4.1 Receipt of Function Present Event

When a managing process receives a Function Present Event, it will determine if it is responsible for managing that function. If so, it will send a Register Request Action. (See Figure 17-1 on page 17-3.)

Note: All managing processes must use the same SAP for reception of the Function Present event. This allows the LAN Station Manager to send one Function Present event to all Managing Processes, thereby reducing traffic. Register Request specifies the SAP value used for future communication between a LAN Station Manager and the Managing Process.

17.4.2 Register Check

The Register Check Action provides the following functions:

- The managing process can query the registration status of a specific station or a group of stations with a particular function. This is useful when a topology change in the network (such as a bridge going down or an CAU reconfiguring) may have caused station manager to reach its lost LANM retry limit and become unregistered.
- The managing process can poll to see what stations exist with a function that it is responsible for managing. This is useful when the managing process attaches to the LAN after the initial Function Present Events are sent by the stations.
- The managing process may perform periodic checks on the functions it is managing to ensure that the registration is not lost. This check may be more frequent for monitoring of critical resources.

If the Register Check is sent to a group of stations, it should be sent by way of a functional address. If the group has a functional address (for example, bridges and CAU), the Register Check should be sent to that address. If the group does not have a functional address, the Register Check should be sent to the resource management functional address (defined as X'C000 0002 0000'). In both cases, it should be single route broadcast.

Response type indicates under what conditions the station should respond:

- if the station manager does NOT have the managing process registered for that group function title and qualifier pair,
- if the station manager DOES have the managing process registered for that group function title and qualifier pair,
- if the station has that group function title and qualifier regardless of whether the managing process is registered or not.

- if the station manager DOES have the managing process registered for that group function title. The qualifier is ignored.
- if the station manager DOES NOT have the managing process registered for that group function title. The qualifier is ignored.
- if the station has that group function title regardless of whether the managing process is registered or not. The qualifier is ignored.

The Register Check may also be directed to an individual station. The station manager returns the registered list for that group identifier and qualifier pair.

When the Register Check is directed at a group of stations, it should place a broadcast value in the managed object instance field of the CMIP action for Register Check. This is necessary because of the lack of scoping and filtering in some implementations of the LAN Station Manager.

This broadcast value is an arbitrary value for the managed object instance for this action. The attribute value should be all *.ones*. This value would indicate that all managed objects of a particular class should respond regardless of the value of the object instance. This is a deviation from CMIP and the only place where it is used is in the registration process between managing process and the LAN Station Manager (within the open system).

Figure 17-4 illustrates the Register Check sequence.

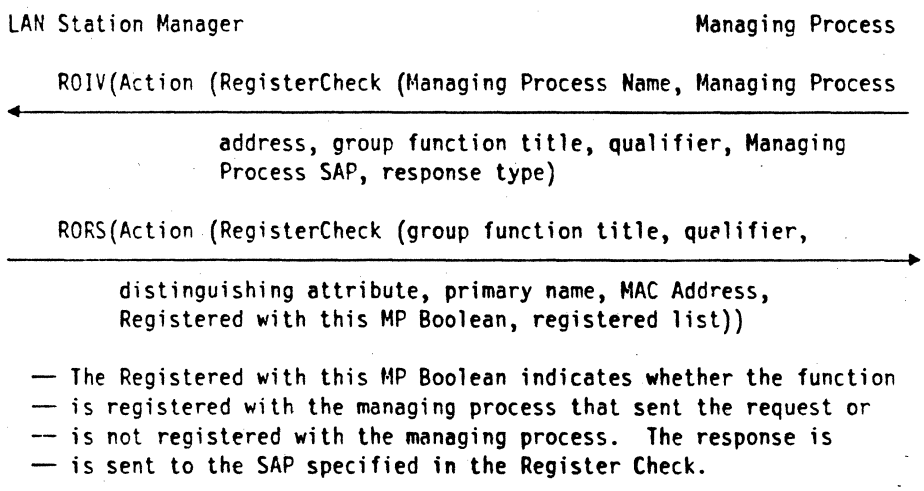


Figure 17-4. Register Check

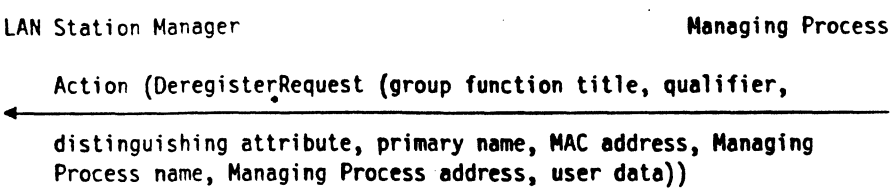
17.4.3 Deregistering

If the managing process receives a Deregister Event, it should remove that function from its table.

If the managing process removes the station from its registration table for reasons other than receiving the Deregister Event, the managing process shall send a Deregister Request Action to that station.

If the managing process is going down, it should send the Deregister Request action to all the stations with which it is registered. It can do this by way of the resource management functional address. Also, the broadcast object instance

value is used when this event is sent. (See 17.4.2, "Register Check" on page 17-9 for details on the broadcast object instance.) The Deregister Request action is unconfirmed.



17.4.4 Multiple Function Registration

In some implementations of LAN Station Manager, there may be several functions to be announced and registered at one time. As defined above, it would be necessary for a separate FunctionPresent Event to be sent for each instance of the group function title that exists. If one managing process wishes to manage all the functions, it would have to send a separate Register Request Action frame for each group function title instance. Such a scenario may occur in a product implementing the LAN Management Server and Bridge Server Objects (see reference {AWP313}). A bridge may have a management server function for each LAN to which it attaches, the bridge function and numerous functions on each adapter.

In order to eliminate traffic on the LAN and processing at both the LAN Station Manager and the managing process, a method of announcing all functions at one time and registering all functions at once as needed.

As a result, two events and two actions have been defined to provide the following functions:

- announce the presence of several functions in one frame (MultipleFunctionPresent Event)
- deregister multiple functions at one time (MultipleFunctionDeregister Event)
- allow a managing process to register several functions at the same address with a single action (MultipleRegisterRequest)
- allow a managing process to deregister several functions at the same address with a single action (MultipleDeregister).

Chapter 18. Common Design for Link-Level Counter Management

18.1 Introduction

This chapter presents a common design for management of link-level counters, independent of the link protocol involved.

18.2 Definition of Terms

agent — a system that plays the agent role: one of the roles defined by OSI Management standards that a system can assume. In the agent role, a system receives commands and applies them to objects. It also emits events based on notifications from objects. Contrast with "manager."

comparison value — the value to which the current count value is compared in the threshold algorithm.

count value — the current contents of a counter. After the count value is set to 0 at counter creation, it is never reset. When the count value exceeds the maximum size of the counter, it wraps to 0 and an event report is sent.

counter set — a group of counters associated with a single layer instance. Whenever an event report is sent for a counter set, data is always included for all of the counters in the set.

denominator threshold — a threshold used as the denominator in the threshold algorithm.

effective reset — the act of adjusting the comparison values for the numerator and denominator counters in a threshold pair, in order to begin a new threshold interval. An effective reset is performed by adding the offset values to the current count values.

initial value managed object — a managed object that stores attribute values to be imported into other managed objects when they are created. Unlike reference objects, i.e., managed objects pointed to by CMIP's CREATE with-reference-object technique, initial value managed objects need not be of the same class as the objects that are created from them.

interval report — a CounterSetReport notification triggered by a threshold on a single counter, rather than by a threshold pair on two counters. Single-threshold triggering of interval reports is treated as a special case of the general triggering of threshold-reached event reports by threshold pairs.

layer instance an entity within a system that implements a layer protocol. Each layer instance is separately addressable in its protocol. Examples of layer instances include token ring LAN MAC instances and LAN LSAPs. The term "layer" here does not necessarily refer to one of the seven OSI layers, since, for example, both token ring LAN MAC instances and LAN LSAPs fall within OSI's layer 2. Layer instances are all derived from the LayerWithCounters managed object class.

manager a system that plays the manager role: one of the roles defined by OSI Management standards that a system can assume. In the manager role, a system initiates commands and receives events. Contrast with "agent."

maximum sample interval a value specified by a manager, indicating the frequency with which a ThresholdControl object *must* evaluate its threshold-pair triggers. An agent is free to sample more frequently than the maximum sample interval if it elects to, but it must sample at least that frequently. See also *sample interval*.

non-resettable counters counters that are never reset to 0. Instead, an effective reset is performed, in order to preserve the current count values for use by multiple managers. All of the counters defined in this document are non-resettable.

numerator threshold a threshold used as the numerator in the threshold algorithm.

offset value an amount added to the current count value of a counter to get a new comparison value.

partial wrap a condition in which a comparison value has wrapped, but the count value has not.

quick wrap a condition in which the count value has wrapped, but a comparison value has not.

report interval the interval between successive CounterSetReport notifications from a ThresholdControl object.

report switch a switch associated with a ThresholdControl object, that determines whether the object is to emit any CounterSetReport notifications. A single switch controls all three types of CounterSetReport notifications: wrap report, threshold-reached report, and deletion report.

sample interval a time interval indicating how often a ThresholdControl object compares its trigger's count values with their comparison values. See also *maximum sample interval*.

threshold interval the time interval between effective resets for a threshold pair.

threshold pair typically, a pair of counters from a counter set, and an associated pair of offset values, that are together used to trigger a CounterSetReport notification. In the special case of an interval report, the "pair" contains only a single counter and a single offset value.

trigger	another term for a threshold pair. A ThresholdControl object's triggers are grouped together in a trigger list.

18.3 Functional Description

This section contains an extensive discussion of the basic counter / threshold design. The emphasis in this section is on the concepts involved. For the templates, see Chapter 21, "Templates" on page 21-1.

18.3.1 Overview of the Basic Design

The purpose of the counter / threshold design is to make it possible for multiple managers to enable statistical reporting by agents based on different triggers. The basic design, presented here and in 18.3.6, "Examples Illustrating the Counter Threshold Design" on page 18-9, is derived from an error-to-traffic ratio (E over T) trigger. It is important to remember, in reading the design, that the terms "numerator" and "denominator" refer to the roles that individual counters / thresholds *would play* in the E over T model. There are no actual numerators and denominators, because there are no actual divisions.

The threshold design typically assigns thresholds in pairs, although in the case of a "pure" interval report, one of the thresholds in the pair is absent. Such a threshold pair is referred to as a *trigger*. A manager wishing to establish multiple triggers for a single counter set will create a *trigger list*, i.e., a set of triggers, each of which operates independently of the others. The model behind the design is an error-to-traffic ratio calculation. One counter is assigned the numerator role in the calculation, i.e., it plays the role of the error counter in E over T. A second counter is assigned the denominator role, i.e., it plays the role of the traffic counter. It is not necessary that the counter in the numerator role actually be an error counter, nor that the one in the denominator role actually be a traffic counter. A manager may assign any counter to either role, and a single counter may be assigned to different roles by different managers, or, for that matter, by a single manager in different threshold pair assignments.

The idea behind the threshold design is a simple one. Suppose the goal is to have an event generated whenever the count in the numerator counter equals or exceeds 25% of that in the denominator counter. Furthermore, to avoid unnecessary event reports for extremely short-term conditions, e.g., when a single error causes both an error counter and an associated traffic counter to have count values of 1, it is specified that the sample must have (in E over T terms) at least 50 errors in 200 traffic PDUs. In this case a threshold pair is set up with an offset of 50 for the numerator (error) counter, and one of 200 for the denominator (PDUs) counter.

With this setup in place, the agent need only check which counter reaches its threshold first:

- If the numerator counter reaches 50 before the denominator counter reaches 200, then the triggering criterion has been met: even if the remainder of the 200 PDUs were error-free, there would have been 50 errors in the 200-PDU sample. Rather than waiting for the full 200 PDUs, however, an event is generated as soon as the numerator counter reaches 50. At this point an effective reset is performed, and a new threshold interval is begun.
- If the denominator counter reaches 200 before the numerator counter reaches 50, then there has been an actual sample of 200 PDUs in which the error-to-

traffic ratio has been less than 25%, and which therefore does not require that an event be generated. The only action required is to begin a new threshold interval.

Complications of detail are introduced by sampling, by the use of non-resettable counters, and by the actions necessitated by the wraps inherent in counters of finite size, but the basic design is that described above.

18.3.2 Key Concepts

Threshold Pair

Thresholds for counters are assigned in pairs, with one counter's threshold playing the numerator role and the other counter's threshold playing the denominator role. For example, a threshold pair might have a threshold on the aborted frames received counter as its numerator threshold, and one on the total frames received counter as its denominator threshold. Relative to this threshold pair, it is also possible to speak of aborted frames received as the numerator counter, and total frames received as the denominator counter.

A threshold pair is also referred to as a trigger, because it triggers the emission of a CounterSetReport notification.

Trigger List

A single manager may desire to have several different threshold-pair triggers operating on a single counter set; it may, for example, want to be notified whenever there are *either* 50 errors in 200 total PDUs, or 50 supervisory PDUs in 1000 total PDUs. A manager specifies multiple triggers by creating a trigger list for the counter set.

So far as its triggering behaviour is concerned, each trigger operates independently of the others. The reports generated as a result of the triggers are, however, related, in that each report communicates the change in the counter set since the previous report for the counter set, event if the previous report was generated as a result of a different trigger.

Threshold Comparison

Each ThresholdControl object maintains a sample interval, indicating how often each of its trigger's comparison values is to be compared with its counter's current value. When a current count value is detected to have reached or exceeded its comparison value, an effective reset for the threshold pair is performed and, if the counter was playing the numerator role in the threshold pair, a CounterSetReport notification is emitted.

Effective Reset

To allow for multiple managers, all counters are non-resettable. When a comparison value is reached or exceeded, and it is thus time to begin a new threshold interval, the counters are not reset to 0. Instead, the numerator and the denominator counters are *effectively* reset: the count values are left as they were, but new comparison values are computed for both counters, by adding to the current count values the offset values specified for the threshold pair.

The point is that for threshold triggering, an effective reset has exactly the same effect as a real reset. In both cases the comparison value C(x) exceeds the count value c(x) by the specified offset value O(x); the only difference is that with a real reset the values of c(x) and C(x) are, respectively, 0 and O(x), while an effective

reset results (ignoring wrap-related complications) in values of c(x) and c(x) + O(x).

Start Values

The current value of a non-resettable counter is not, by itself, especially useful: what is of interest is the *recent* behaviour of the counter, not its behaviour since its layer object was initialized. In order that this recent behaviour can be reported in a CounterSetReport notification, a ThresholdControl object maintains two types of start values:

- *Interval start values* are maintained for every counter in the counter set. These values are updated every time a CounterSetReport notification is emitted, and thus cover the change in a counter between CounterSetReports. If a manager could be guaranteed to receive every CounterSetReport emitted by a ThresholdControl object, then interval start values would be unnecessary, since the interval start values reported in one report are identical to the current values reported in the previous one. Since, however, this cannot be guaranteed, interval start values are included in each report, to make each such report self-contained.
- *Effective reset start values* are maintained for the [one or] two counters that make up a threshold-pair trigger. These values are updated whenever the trigger performs an effective reset, whether or not a CounterSetReport is emitted. They are included in any CounterSetReports resulting from that threshold pair, to indicate how much the trigger counters increased during the threshold interval that led to the report. Note that this information cannot be inferred from these counters' interval start values, which are also reported in the CounterSetReport, since, in general, report intervals and threshold intervals are not synchronized. See "Threshold Interval and Report Interval" on page 18-7 for an example that further illustrates this point.

See 18.3.10, "Processing of Start Values by a Managing Process" on page 18-21 for a discussion of exactly how a manager uses the information contained in a CounterSetReport to determine a counter's recent behaviour.

Partial Wrap

When a comparison value wraps, a flag is set indicating that the threshold has wrapped and the counter has not. This condition is termed a partial wrap. So long as a threshold is in partial wrap, the count value *must* be less than the comparison value (in absolute terms), since the comparison value is one "pass" ahead of the count value. A straight arithmetic comparison in this case would typically yield the wrong answer, however, since a comparison value that has just wrapped will ordinarily be less than a count value that is about to. Thus so long as a threshold is in partial wrap, arithmetic comparison is suspended, and the partial wrap flag itself is taken as an indication that the count value is less than the comparison value.

Once the count value catches up with the comparison value, i.e., gets back on the same "pass" through the counter, the partial wrap flag is reset and ordinary arithmetic comparisons are resumed.

Quick Wrap

If the count value in incremented by a large number in a single operation, or it is incremented a number of times during a sample interval, the count value may wrap prior to a comparison value. This condition, termed a quick wrap, is the inverse of a partial wrap. Once again a flag is needed, because a straight arithmetic comparison would give the wrong answer: if the count value has just wrapped and a com-

parison value is about to, the count value will typically be less than the comparison value, even though it is greater in absolute terms, since it is one "pass" ahead.

The quick wrap flag is set only long enough for the comparison to take place. Then, since the count value exceeds the comparison value, an effective reset is performed and the comparison value gets back on the same pass with the count value.

Role

A threshold performs one of two roles, numerator or denominator.

Report Switch

A switch that determines whether CounterSetReport notifications are to be emitted by a ThresholdControl object. A single switch controls generation of all the reports emitted by a ThresholdControl object, i.e., deletion reports, wrap reports, and threshold-reached reports for each of the triggers in the object's trigger list.

18.3.3 Event Reports

There is one type of event report defined, the CounterSetReport, to report several different conditions; layer instance deletion, deletion of the Threshold Control object itself, counter wrap, or Threshold reached. The CounterSetReport is emitted by the ThresholdControl object, and always contains the same core data, in the same format, for the counter set being reported upon. In the case of wraps and threshold-reached conditions, it also contains additional information related to these specific conditions.

The MSOBV-CounterSetReport behaviour template in Chapter 21, "Templates" on page 21-1 indicates the data included in each of the types of CounterSetReports. See the abstract syntax definitions in Chapter 23, "ASN.1 Definitions" on page 23-1 for more details on exactly what this data is.

18.3.4 Types of Intervals Defined by the Algorithm

In order to avoid confusion, it is important to distinguish very carefully among the three types of time intervals that play a role in the algorithm. The following sections seek to do this.

Sample Interval

The sample interval accommodates periodic sampling of counters. A single value is used for an entire ThresholdControl object, for all of the triggers in the object's trigger list. This value determines how often the counters in each trigger's threshold pair are compared against their comparison values to see if they have reached them. Since it is associated with the threshold pairs in the trigger list, the sample interval applies only to the [one or] two counters that are actually in each pair; it has nothing to do with any other counters that may be in the counter set.

When a ThresholdControl object is created or modified, a manager specifies a *maximum* sample interval to be used by the object. An agent is free to use this specified interval, or any shorter one, for the operation of the object.

Threshold Interval and Report Interval

These two intervals are best discussed together. A threshold interval is defined to be the interval between effective resets; thus it is a per-trigger interval. By contrast, a report interval, being the interval between CounterSetReport notifications, is a per-ThresholdControl object interval. The following figure and text explain how these two types of intervals are related.

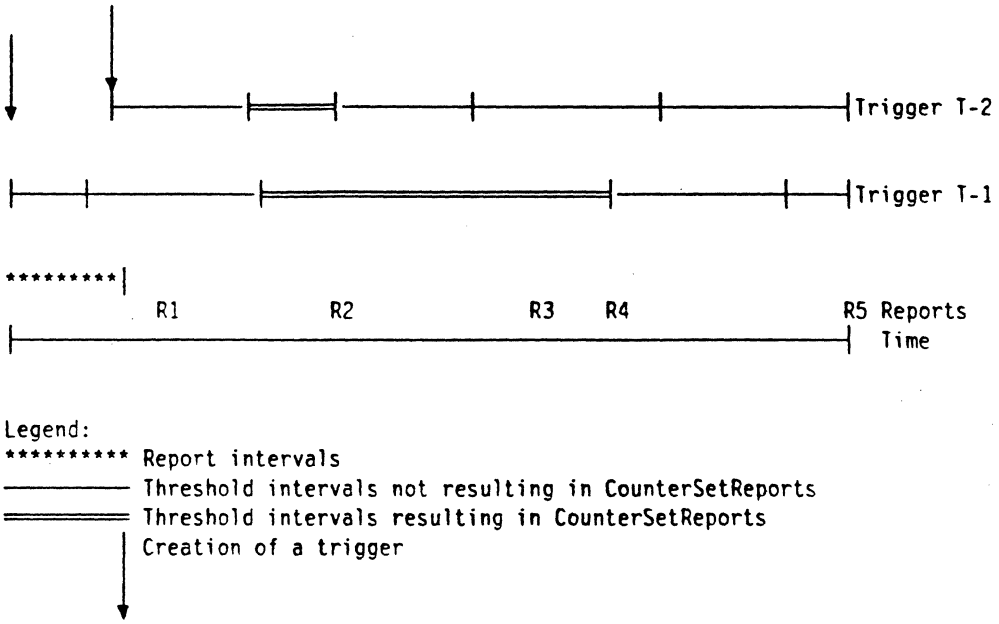


Figure 18-1. Relationship between Threshold Intervals and Report Intervals. Everything shown in this figure applies to a single ThresholdControl object.

Figure 18-1 illustrates the relationship between threshold intervals and report intervals. One key aspect of this relationship is the way in which the updating of start values works. After initialization at the creation of the ThresholdControl object, the interval start values for all the counters in the set are updated every time a report is emitted. In addition, each trigger maintains effective reset start values for the [one or] two counters involved in its threshold pair. Effective reset start values are updated every time there is an effective reset for the pair, whether or not a report is emitted.

The figure shows the following sequence of events:

Object creation The ThresholdControl object is created, with trigger T-1 in its initial trigger list. This may have occurred at the same time that the layer instance and its contained counter objects were created, or it may have occurred later. Interval start values for the counter set are initialized to the current values of the counters at the time of the ThresholdControl object's creation. Effective reset start values for T-1's counters are also initialized, to the current values of these two counters.

Effective reset for T-1 Trigger T-1 performs an effective reset, without emitting a CounterSetReport notification. T-1's effective reset start values are updated.

Creation of T-2 Trigger T-2 is created, via, a CMIP command to add it to the set-valued TriggerList attribute. T-2 does not maintain a separate set of interval start values—there is one set for the entire ThresholdControl object. Thus the only variables that need to be initialized are the effective reset start values for the counters in T-2's threshold pair.

R1 emitted CounterSetReport R1 is emitted. Since this report is associated neither with a trigger nor with a deletion, it must be a wrap report. R1 contains the current values of the counters in the counter set, as well as their interval start values. Thus the receiving manager can determine by how much the counters have incremented since the ThresholdControl object was created.

When R1 is emitted, the ThresholdControl object updates the interval start values; they now match the current count values that were reported in R1.

Effective reset for T-2 Trigger T-2 performs an effective reset, without emitting a CounterSetReport notification. T-2's effective reset start values are updated.

Effective reset for T-1 Trigger T-1 performs a second effective reset, again without emitting a CounterSetReport notification. T-1's effective reset start values are updated a second time.

R2 emitted T-2's triggering criterion is satisfied, so a CounterSetReport is emitted. This report contains interval start values for the entire counter set, covering the interval since R1 was emitted. Additionally, it contains effective reset start values for the two triggers in T-2's threshold pair, covering the shorter interval since T-2's last effective reset.

When R2 is emitted and T-2 performs its effective reset, both the interval start values for the entire counter set and T-2's effective reset start values for the counters in its threshold pair are updated.

Effective reset for T-2 Trigger T-2 performs an effective reset, without emitting a CounterSetReport notification. T-2's effective reset start values are updated again.

R3 emitted CounterSetReport R3, another wrap report, is emitted, and the interval start values are updated.

R4 emitted This time it is T-1's triggering criterion that is satisfied, causing a CounterSetReport to be emitted. In addition to interval start values for the entire counter set, covering the interval since R3 was emitted, this report contains effective reset start values for the two triggers in T-1's threshold pair. These effective reset start values cover the interval all the way back to T-1's last effective reset, prior to R2.

When R4 is emitted and T-1 performs its effective reset, both the interval start values for the entire counter set and T-1's effective reset start values for the counters in its threshold pair are updated.

Effective reset for T-2 Trigger T-2 performs an effective reset, without emitting a CounterSetReport notification. T-2's effective reset start values are updated again.

Effective reset for T-1 Trigger T-1 performs an effective reset, without emitting a CounterSetReport notification. T-1's effective reset start values are updated again.

R5 emitted R5 is a deletion report, emitted when the ThresholdControl object is deleted. (This may be a deletion just of the ThresholdControl object, or it may be a side effect of the deletion of its containing layer instance; the tag in R5 indicates which of these has taken place.) R5 contains interval start values covering the interval since R4 was emitted.

18.3.5 Algorithm for Counter Thresholds

See MSOBV-Counter and MSOBV-ThresholdControl in Chapter 21, "Templates" on page 21-1 for the exact definition of the algroithm for counter thresholds.

18.3.6 Examples Illustrating the Counter Threshold Design

This section contains a number of examples illustrating how the counter / threshold design actually works. Examples typically show the state of two objects before and after a specified event takes place:

- a counter, which happens to have a maximum size of 299
- a table entry for this counter, for a threshold x; there will be an entry in this table for each threshold currently defined for the counter.

Except for the special case of "pure" interval reports, thresholds always exist in pairs. Thus whenever an entry in Counter 1's table says

```
ID = x
pointer = Counter 2
```

there will be a corresponding entry in Counter 2's table saying

```
ID = x
pointer = Counter 1.
```

This is the mechanism by which one counter's reaching its threshold causes an effective reset of both counters in the pair.

18.3.7 Threshold Creation

The following examples illustrate the creation of a new threshold for a counter. Ordinarily thresholds are created in pairs, in which case the actions shown here are performed for each of two counters within a counter set. If, however, the requesting manager is enabling a "pure" interval report, only a numerator threshold is created.

There are two cases, depending upon whether adding the offset to the current count value results in a wrap. Assume for both cases that the following values are specified for the new threshold:

- **ID = x**: This is an identifier associated with the threshold *pair*, i.e., this threshold shares the value with its peer threshold on another counter. The identifier is used to find the other threshold in the pair when an effective reset is to be performed.

- **Pointer = Ci**: This is a pointer to the other counter in the threshold pair. For pure interval reporting, this pointer is null for the numerator threshold. In other words, since there is no counter playing the denominator role for pure interval reporting, the pointer *to* the denominator counter from the numerator threshold is null.

  This value is deduced from the threshold-creation command from the manager, by noting which other counter, if any, is identified in the command.

- **Role = Numerator**: This value is specified in the threshold-creation command from the manager.

- **Offset = 50**: This value is specified in the threshold-creation command from the manager.

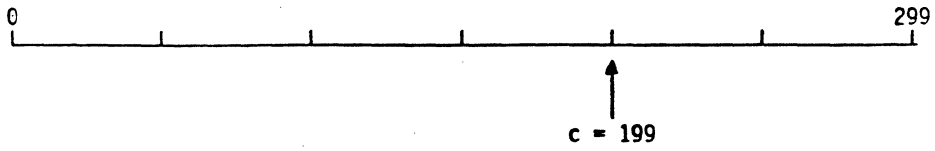
- **Event Report Switch = on**: This value applies only to a numerator threshold; it is specified in the threshold-creation command from the manager. It is hard to imagine why a manager would ever set this switch to "off" on an initial threshold-creation command. It might, however, be useful to turn the switch off for a short time later, if the manager needs to turn off event reports temporarily, but does not wish to cancel the threshold setup altogether.

  A report switch applies to an entire ThresholdControl object. Thus the "SW = on" in these examples should be interpreted to mean that the switch is set to "on" for the ThresholdControl object that contains the trigger shown.

Note that the partial wrap and quick wrap flags are never specified by the manager. Instead, they are set by the agent, depending on the behaviour of the counter itself.

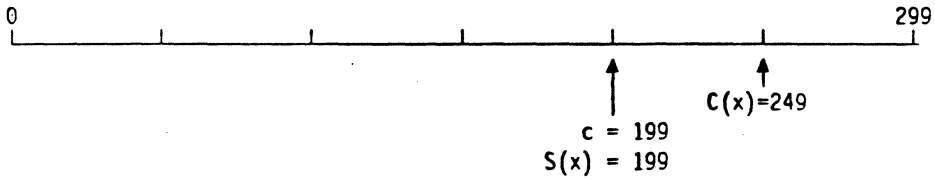
c + O(x) < = Max Value

Before:



ID	Pointer	Role	Offset	PW	QW	SW

After:

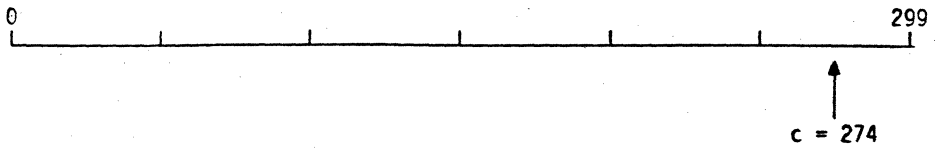


ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

The partial wrap flag PW for a threshold indicates that the comparison value is one "pass" ahead of the count value, and the quick wrap flag QW indicates that the count value is one "pass" ahead of the comparison value. Since both values are on the same "pass" in this case, neither flag is set on.

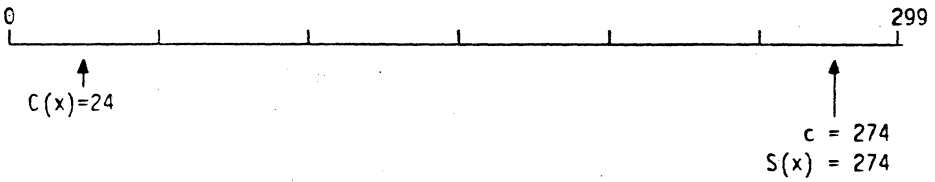
c + O(x) > Max Value

Before:



ID	Pointer	Role	Offset	PW	QW	SW

After:



ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	on	off	on

Since the comparison value is one "pass" ahead of the count value, the partial wrap flag PW is set on.

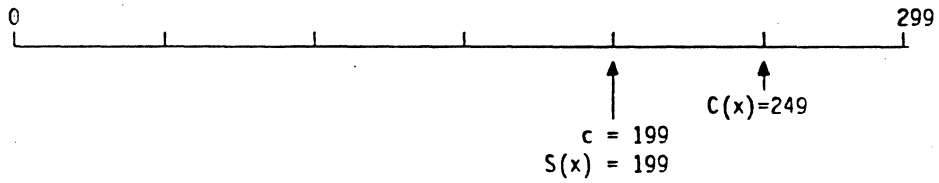
18.3.8 Actions Taken When a Counter is Incremented

The following examples illustrate the different sequences of events that can take place when a counter is incremented. The focus of the examples is on the interactions between the count value c, the comparison value C(x), the partial wrap flag PW(x), and the quick wrap flag QW(x). Since the examples show a numerator threshold with its event report flag set on, an event report will be generated whenever the test for threshold reached turns out true. The only difference, though, if the threshold were playing the denominator role, or if it were a numerator threshold with the event report flag set off, is that there would be no event report: everything else would be exactly the same.

Assume in all of the examples that follow that i = 25, i.e., the counter is being incremented by 25; the offset remains 50.

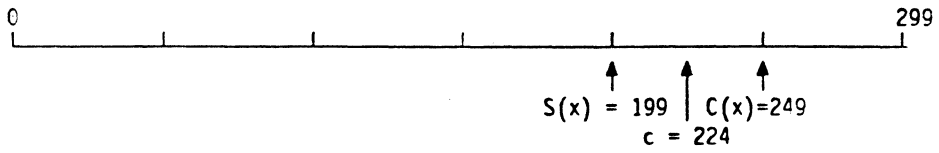
c + i < C(x) <= Max Value

Before:



ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

After:

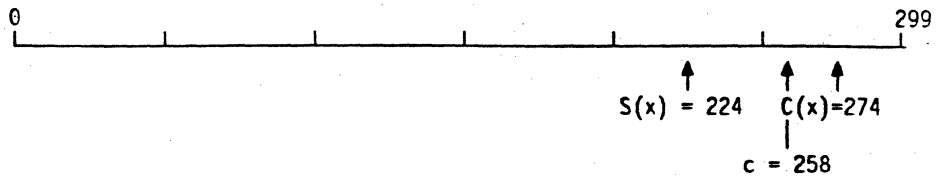


ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

This is the simplest case, where (1) the threshold is not in a partial wrap condition to begin with, and (2) even after it is incremented, the count value is less than the comparison value.

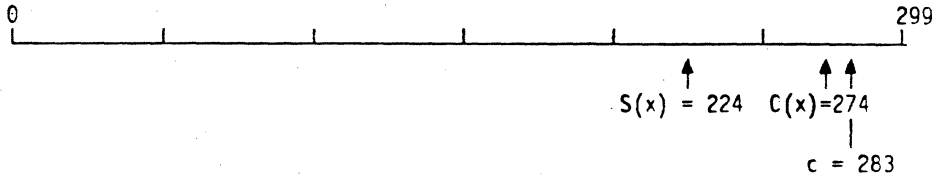
C(x) < = c + i < = Max Value

Before:



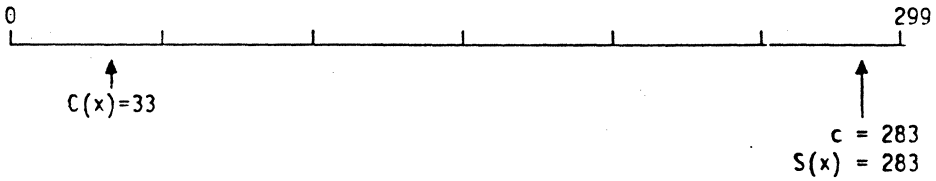
ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

Intermediate:



ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

After:

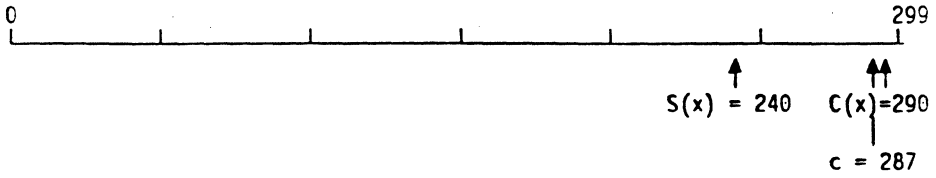


ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	on	off	on

Since, at the time of the comparison, the count value 283 exceeded the old comparison value 274, an effective reset was performed. Now the comparison value is one "pass" ahead of the count value, so the partial wrap flag PW is set on.

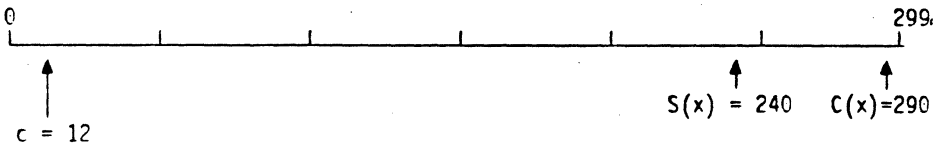
C(x) < = Max Value < c + i

Before:



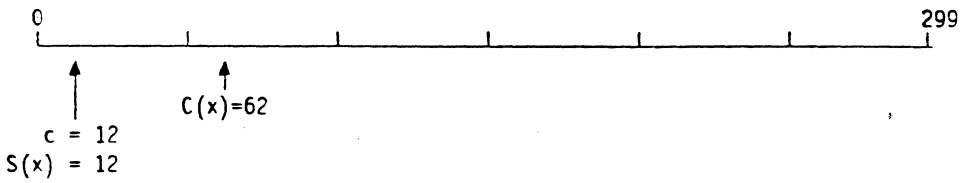
ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

Intermediate:



ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	on	on

After:



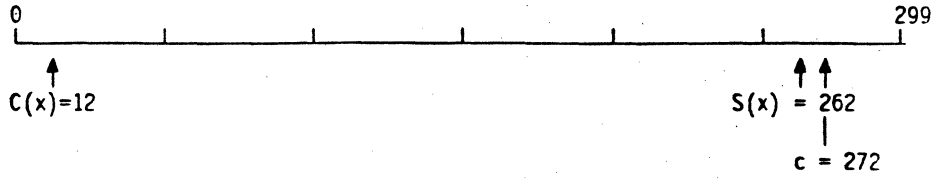
ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

Since, at the time of the comparison, the quick wrap flag QW was set on (because the count value had wrapped but the comparison value had not), an effective reset was performed. After the effective reset the count value and the comparison value are back on the same "pass"; thus the quick wrap flag QW is set back off.

One additional event takes place in this example. Because the counter has wrapped, a wrap report must be generated.

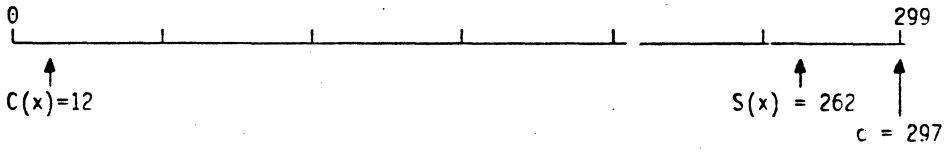
c + i <= Max Value, Threshold in Partial Wrap

Before:



ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	on	off	on

After:

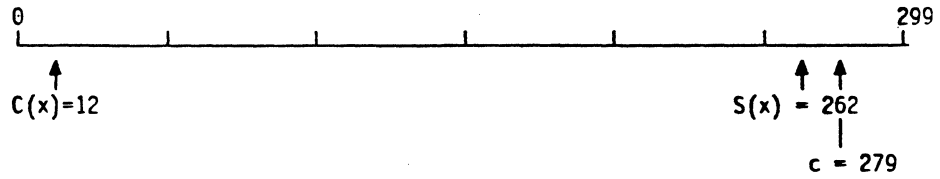


ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	on	off	on

Since the count value does not wrap after it is incremented, nothing changes: the comparison value is still 12, and the threshold is still in partial wrap.

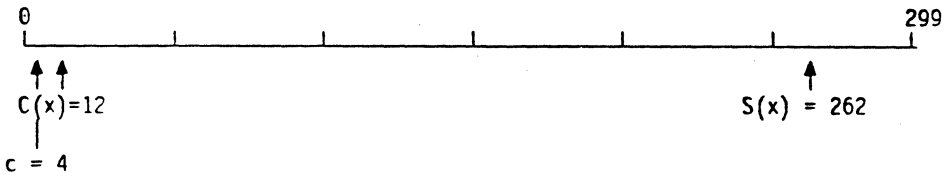
Max Value < c + i < Max Value + C(x), Threshold in Partial Wrap

Before:



ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	on	off	on

After:



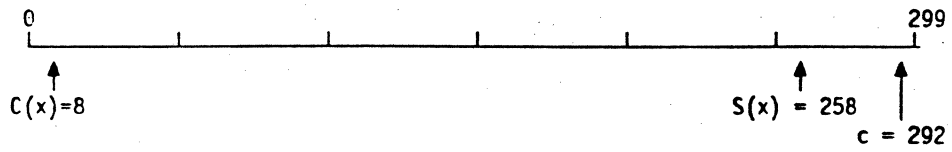
ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

Several things happen in this case:

- Since the count value wraps, a wrap report is sent.
- The partial wrap flag PW is set to off, since the count value and the comparison value are now back on the same "pass."
- Upon comparison, the count value is less than the comparison value, so no event report is sent and no effective reset is performed.

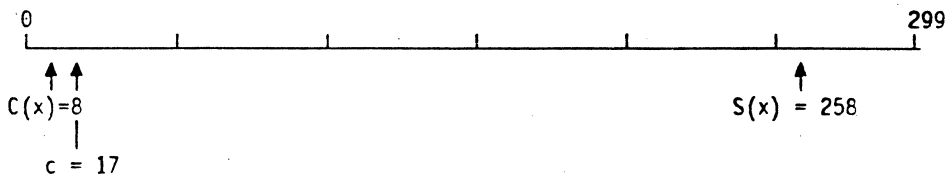
C(x) <= c + i, Threshold in Partial Wrap

Before:



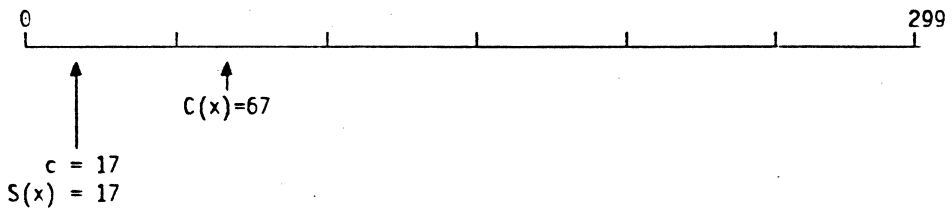
ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	on	off	on

Intermediate:



ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

After:



ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

Several things happen in this case:

- Since the count value wraps, a wrap report is sent.
- The partial wrap flag PW is set to off, since the count value and the comparison value are now back on the same "pass."

- Upon comparison, the count value exceeds the comparison value, so an event report is sent and an effective reset is performed.

Note that in this case two reports are sent: one because the counter wrapped, and one because a numerator threshold was reached and the event report flag was turned on.

18.3.9 Effective Reset Triggered by a Paired Counter

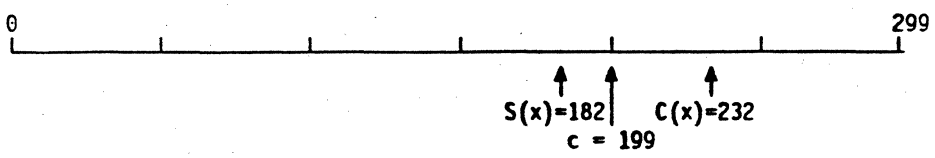
Several of the preceding examples illustrated effective reset of a counter that had reached or exceeded one of its comparison values. The two examples that follow illustrate the "other" half of the effective reset, that of the counter that did not reach its threshold. These examples are very similar to those in 18.3.7, "Threshold Creation" on page 18-9. In both groups a counter is having its comparison value for a threshold set to the current count value c plus the threshold's offset value O(x). The only difference between these cases and the earlier ones is that since the threshold already exists, the "permanent" values *ID*, *Pointer*, *Role*, *Offset*,and *Switch* are already specified.[1]

There are two cases, depending upon whether adding the offset to the current count value results in a wrap. Assume as before that the offset value O(x) = 50.

[1] A comparison of the two pieces of pseudo-code in Chapter 21, "Templates" on page 21-1 for creating a threshold (i.e., adding a trigger) and for effective reset reveals one other apparent difference. For threshold creation the partial wrap flag is set on whenever the comparison value wraps, while for effective reset there is an additional test to see whether the quick wrap flag is already on. Since, however, the quick wrap flag is on only while a counter that was incremented is being updated, it will never be on for the effective reset of the "other" counter, i.e., the one that was not incremented. Thus the processing of the wrap flags in the two cases really is identical.

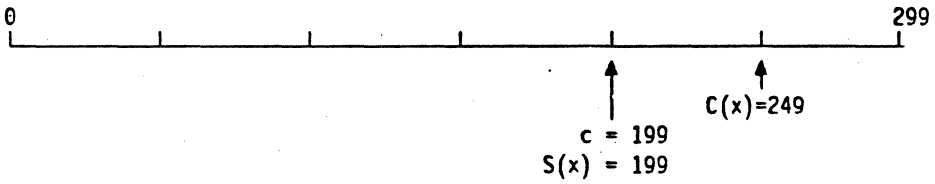
c + O(x) < = Max Value

Before:



ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

After:

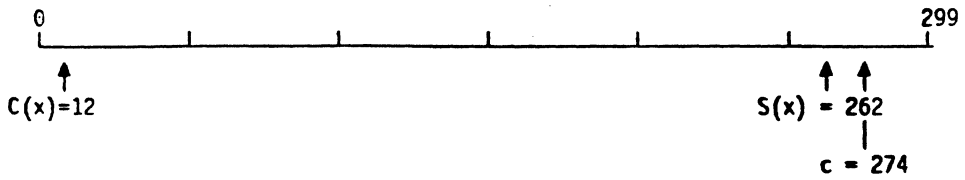


ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	off	off	on

The partial wrap flag PW for a threshold indicates that the comparison value is one "pass" ahead of the count value, and the quick wrap flag QW indicates that the count value is one "pass" ahead of the comparison value. Since both values are on the same "pass" in this case, neither flag is set on.

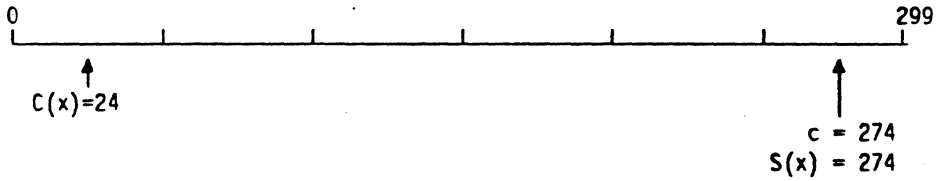
c + O(x) > Max Value

Before:



ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	on	off	on

After:



ID	Pointer	Role	Offset	PW	QW	SW
x	Ci	Numerator	50	on	off	on

In this example the partial wrap flag PW happened to be on prior to the effective reset, but this had no effect on the operation. Exactly the same final state would have resulted if the threshold had not been in partial wrap to begin with.

18.3.10 Processing of Start Values by a Managing Process

Interval start values and effective reset start values are included in CounterSetReports to allow a manager to determine how much a counter has incremented during the interval being reported on. If counters never wrapped, the manager would always be able to get the right answer by simply subtracting a counter's interval or effective reset start value from its current value. Since counters do wrap, however, simple subtraction is not sufficient.

The general formula a manager needs to use to determine how much a counter has incremented is

```
CurrentValue - StartValue + n*(MaxValue+1)
```

where n is the number of times the counter has wrapped. The difficulty lies in determining what the correct value for n is in all cases.

The use of CounterSetReports every time a counter wraps greatly simplifies the problem. So long as it is guaranteed that a wrap report (i.e., a CounterSetReport in which the tag identifies a counter wrap as the reason for the report) is sent

immediately whenever a counter wraps, the problem is solved for interval start values. A manager uses the formula above with:

wrap reports:

- n = 1 for the counter(s) that have wrapped
- n = 0 for all other counters in the set

all other types of CounterSetReports:

- n = 0 for all counters in the set

Note that a wrap report explicitly includes the maximum values for any counters that have wrapped, to let the manager make this calculation.

For this scheme to work, the wrap report must always be sent prior to any threshold-reached reports that might have been triggered by the same event that caused the counter to wrap. This allows the wrap report to get the interval start values back on the same pass with the current count values, and thus to insure that n = 0 yields the correct answer when the threshold-reached report is processed.

The situation is almost the same with effective reset start values, except that there are no wrap reports involved. Here it is necessary to make two additional assumptions:

- the offset specified for a counter when its threshold pair is defined is less than the counter's maximum size.
- a counter never increments by more than its maximum size during a single sample interval.

Given these two assumptions, the manager uses the following procedure to interpret effective reset start values:

- If the effective reset start value is $\leq$ the current value, then n = 0.
- If the effective reset start value is > the current value, then n = 1.

The maximum values for the trigger counters are always included in a threshold-reached report, in case the manager needs them for this calculation.

18.3.11 Frequency of Wrap Reports

Since the preceding discussion has highlighted the role of wrap reports in the processing of interval start values, a natural question to ask is, how often will wrap reports be occurring? There are three different answers to this question, depending on the counter involved.

Error Counters

Error counters are not likely to wrap ever. Since counters are defined to be strictly increasing, it is obvious that any counter, if it stays around long enough, will eventually wrap.

The Time Counter

The units defined for the time counter are milliseconds, and the minimum size allowed for this counter is 4 octets. A straightforward division yields the answer that a 4-byte time counter will wrap in just under 50 days. (It was a similar calculation that ruled out a 2-octet time counter, since it would wrap roughly once a minute.) Thus if a layer instance stays up continuously for 50 or more days, a manager monitoring the instance should expect to see a wrap report for the instance's time counter.

18.4 Class Inheritance Structure

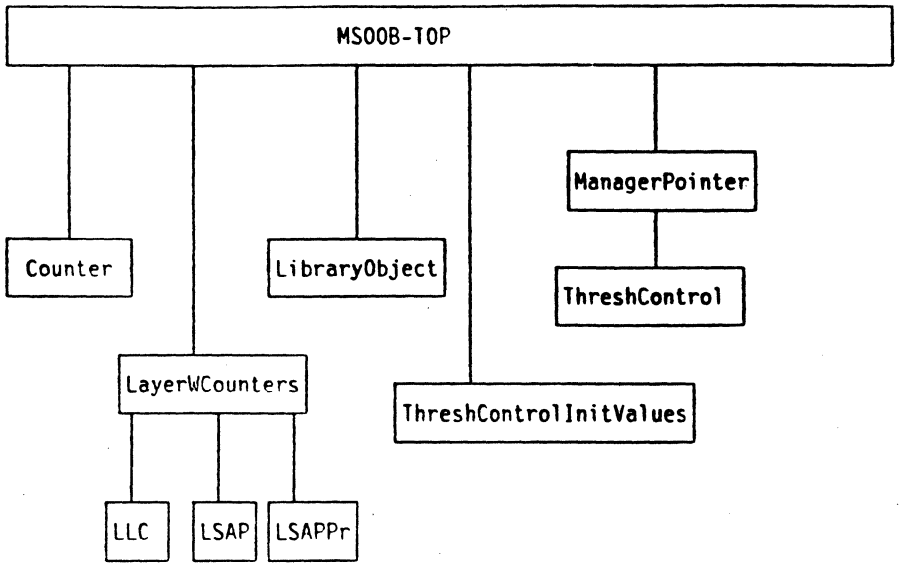


Figure 18-2. Class Inheritance

18.5 Containment Hierarchy

Figure 18-3 illustrates a portion of a general containment hierarchy. In an actual agent system the "Layer Instance" would be replaced by a specific protocol's layer instance, e.g., LAN's LSAPPAIRENTITY.

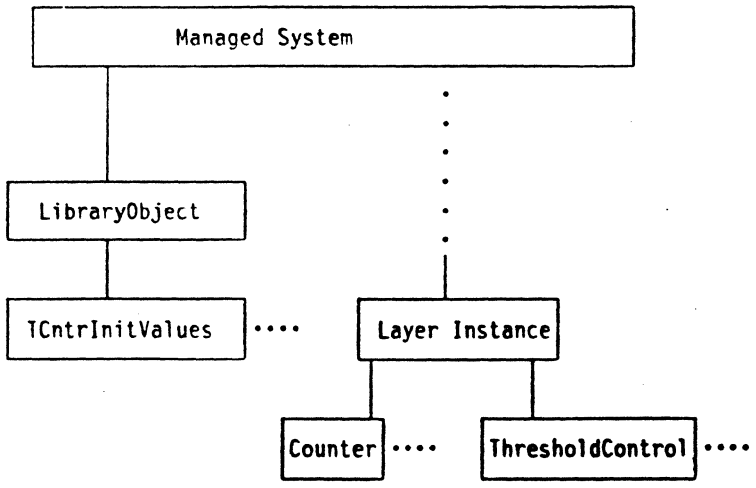


Figure 18-3. The Containment Hierarchy

Chapter 19. Object Definitions

This chapter describes the LAN object classes and a naming scheme for identifying instances of these objects.

The formal object definitions use a set of templates provided by the ISO standard, "Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects." These templates have labels which refer to other templates or to data types and values contained in ASN.1 modules. The templates and the ASN.1 modules together provide the information necessary to construct and parse all the ROSE/CMISE/LAN PDUs in this architecture.

The naming scheme defines the naming attributes for each of the object classes.

The templates for the objects, attributes, notifications, actions, behaviors, and name bindings are located in Chapter 21, "Templates" on page 21-1.

19.1 Object Classes

This architecture specifies the following object classes:

- Environment
- CAU
- Attachment Module
- Token-Ring Layer 1
- Token-Ring Layer 2 - MAC
- LAN Layer 2 - LLC
- LSAP Entity
- LSAP Pair Entity
- Resource Management
- Name Management

19.2 Inheritance

An object template contains a specification of the object classes from which the object is derived. If object class *x* is derived from object class *y*, all of the attributes, operations, and notifications defined as part of object class *y* are inherited by object class *x* and are subject to the same CMIP operations as the attributes, operations and notifications that are explicitly defined as part of object class *x*. Top is a class defined by ISO Management standards as the starting point for all managed object definitions.

Figure Figure 19-1 on page 19-2 illustrates the inheritance tree.

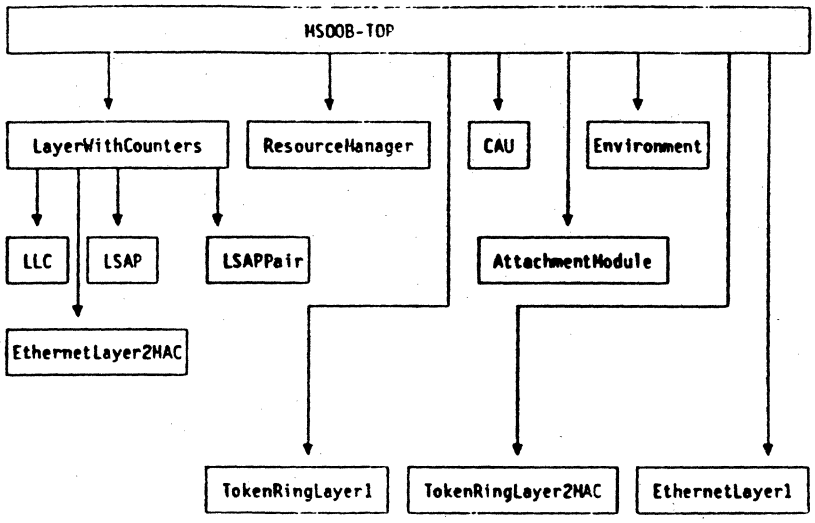


Figure 19-1. LAN Station Manager Inheritance Tree

19.3 Object Naming

This section specifies the naming scheme for the managed objects defined by this architecture. The scheme is based on *Structure of Management Information* and *Guidelines for the Definition of Managed Objects* standards. The formal definition of the name bindings are located in Chapter 21, "Templates" on page 21-1

The SMI naming scheme is hierarchical; the hierarchy is defined by arranging the managed object classes into a "containment tree." In this terminology, instances of a subordinate object class are "contained" in its superior object class. Most of the intuitive properties of physical containment are followed; an object cannot be partially contained in another object, an object cannot be simultaneously contained in two objects unless one of them is itself contained in the other, etc. For example, a network contains nodes and links; nodes contain equipment and software, etc. An object class may appear in more than one place in the tree, for example, instances of counter objects may be contained in different layer objects. The containment hierarchy may also be recursively defined.

For each object class in the containment tree, an attribute in that class is selected for the purpose of naming; different values of the naming attribute distinguish instances of the class within the scope of its superior class. For example the LSAP object class has an attribute, LSAPId. If LSAP is contained in Token-Ring Layer 2 MAC, instances of LSAP are distinguished by assigning unique LSAP values. Note that two LSAPs may have the same number so long as they are not "contained" within the same Token-Ring Layer 2 MAC object.

The name for an object instance is constructed by concatenating the name attribute values from the tree root to the object of interest. These names are called ***distinguished names***. Each name component of the distinguished name is called a ***relative distinguished name***. An RDN consists of a set of ***attribute value assertions***.

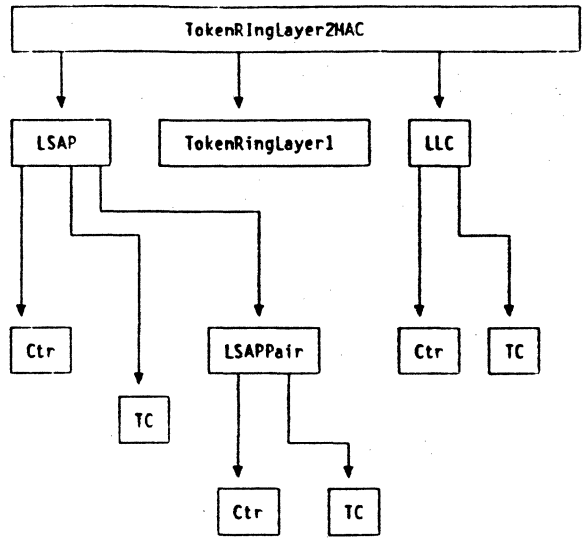
An AVA is a sequence of the attribute ID of the naming attribute and the value of the naming attribute. The RDN and AVA constructs are borrowed from the Directory Services standards. OSI Management does not allow the full generality for RDNs that Directory defines. Specifically, OSI Management allows only one AVA per RDN.

The containment hierarchy is used for NAMING, not for modelling physical or logical relationships between object classes. Physical and logical relationships between object classes are useful properties to consider in defining the containment hierarchy.

The LAN Object Instances are named by the local distinguished name. This means that the object instance in CMIP frames is not the globally distinguished name with the full sequence of AVA from the managed system name down to the name of the object emitting the CMIP frame. So, in the case of the LAN objects defined by this architecture, the object instance name begins with the highest relevant piece of the containment tree described on the following pages. The CAU instance name begins with the AVA for access unit Id. Any object which falls below MAC in the containment hierarchy will begin its instance name with MAC address.

The formal name bindings for these objects can be found in Chapter 21, "Templates" on page 21-1. Because the structure of the tree beyond the LAN environment is unknown by the LAN objects, the superior objects for the highest part of the LAN containment tree is unknown. For this reason, there are no naming bindings for the TokenRingLayer2MAC, CAU, and Resource Management objects.

19.3.1 Description of the Containment Hierarchy (Token-Ring MAC Branch)



Key:

Ctr: Counter
TC : Threshold Control

Figure 19-2. Containment Hierarchy (Token-Ring MAC Branch)

19.3.2 Description of the Containment Heirarchy (Ethernet)

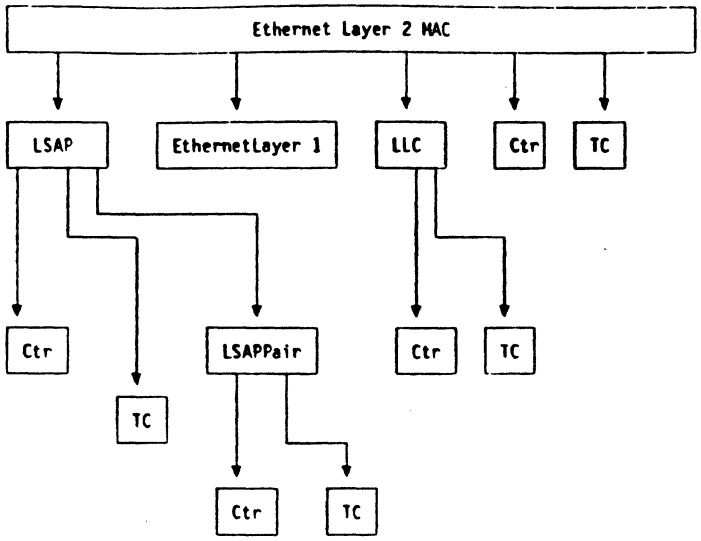


Figure 19-3. Description of the Containment Hierarchy (Ethernet)

19.3.3 Description of the Containment Hierarchy (Resource Manager and CAU branch)

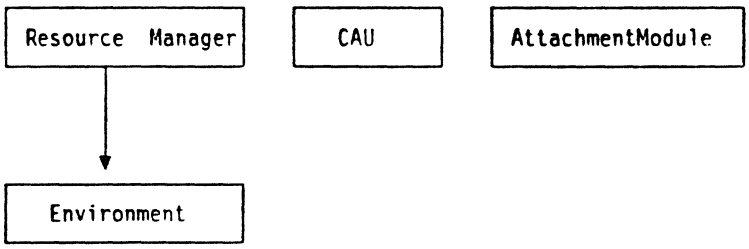


Figure 19-4. LAN Station Manager Containment Hierarchy

An explanation and supporting rationale for this containment hierarchy follows. The name binding templates define what attribute is used in defining the AVA.

CAU object class represents the controlled access unit. The access unit ID is used to uniquely name the CAU instance.

The Attachment Module object class represents the attachment modules that are contained in the Controlled Access Unit. It is named by the Attachment Module name, which is the access unit ID followed by the attachment module number.

Resource Management represents the LAN station manager. It is named by its primary name.

Environment is an object that represents physical equipment. It is subordinate to the resource management object and is named by the EnvName attribute.

Token-Ring Layer 2 MAC represents the media access control sublayer. The individual MAC address is used to name the Token Ring Layer 2 MAC instance.

Token-Ring Layer 1 represents the physical layer of Token-Ring. A Token-Ring Layer 1 instance is uniquely identified by the attribute PHYName.

Ethernet Layer 2 MAC represents the media access control sublayer. The individual MAC address is used to name the Ethernet Layer 2 MAC instance.

Ethernet Layer 1 represents the physical layer of Ethernet. An Ethernet Layer 1 instance is uniquely identified by the attribute EthernetLayer1Name.

The name management part of LAN Station Manager is modelled by the Name Management object class. The MAC address is used to name the name management object instance, so the object class is placed subordinate to Token-Ring Layer 2 MAC.

The LAN Layer 2 LLC object class represents the LLC Entity. The LLC entity is identified by the MAC address. Thus, the LAN Layer 2 LLC object class is placed subordinate to the Token-Ring Layer 2 MAC object class.

LSAP Object Class represents the SAPs in the LLC Entity. A LSAP ID represents a particular instance of the LSAP. The Token-Ring Layer 2 MAC address serves to uniquely identify the SAP, so the LSAP Object Class is subordinate to Token-Ring Layer 2 MAC object class.

The LSAP Pair Object Class represents the end of a link connection. There may be many LSAP Pairs per LSAP, so the LSAP Pair Object Class is subordinate to LSAP Object Class. The remote MAC address and remote SAP are used as the naming attribute.

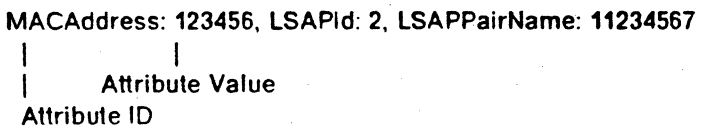


Figure 19-5. Example naming structure

The naming scheme is defined by a set of bindings, where each binding specifies a naming attribute in a subordinate object class with a superior object class. This is equivalent to specifying the containment tree by enumerating the mother/daughter pairs node pairs.

Chapter 20. Managed Object Overview

The following describes the information managed by LAN Station Manager for a Token-Ring station and its access type (get and/or replace):

- **Environment**

  EnvName
  : specifies the naming attribute for the Environment object class (get)

  location
  : specifies the location of the box in which station manager resides (get-replace)

  machineType
  : specifies the type of machine in which the station manager resides (get)

  serial number
  : specifies the serial number of the machine in which the station manager resides (get)

  user defined data
  : (get)

- **Token-Ring Layer 1**

  TRLayer1Name
  : specifies the naming attribute for this managed object class (get)

  access unit ID
  : specifies the unique identifier of the access unit to which the MAC is attached. The CAU will set this value in a station via the CMIP SET command once a station has attached to a lobe.

  access unit number
  : specifies the user-defined value of the access unit or repeater. If it is not entered by the user, the value is NULL. Otherwise, the access unit is a printable string. This attribute is different from the access unit id. The access unit ID is the managed object instance name for an intelligent access unit that is obtained from the MAC address. The access unit number is a value used for all access units and repeaters that is assigned by the customer. (get-replace)

  segment data rate
  : specifies the data rate of the segment on which the MAC resides (get)

  lobe receptacle number
  : specifies the identification of the lobe on which the adapter resides The CAU will set this value in a station via the CMIP SET command once a station has attached to a lobe. (get-replace)

  wall faceplate label
  : specifies the wall connection to the access unit (get-replace)

AM Number

specifies the module that attaches the adapter to the access unit The CAU will set this value in a station via the CMIP SET command once a station has attached to a lobe. (get-replace)

adapter number

specifies the adapter number for this particular adapter out of all the adapters in the box (get)

ringUtilization

specifies a conditional package that gives the calculated ring utilization (This is optional because only some adapters have the ability to calculate this number) (get)

- **Token-Ring Layer 2 MAC**

individual MAC address

specifies the address of the device on the segment (get)

universally-administered address

specifies the default MAC address shipped with the device (get)

functional address

specifies the subset of group addresses that this station has active. (get-replace)

group address

specifies the group addresses that the device has opened. (get-replace)

upstream neighbor address

specifies the MAC address of the station's upstream neighbor (get)

segment number

specifies the segment number on which this MAC resides (get replace)

microcode level

specifies the microcode level of this device (get)

product instance ID

specifies the identifier used to uniquely identify the hardware product instance of the attached product. (get)

AccessPriority

specifies the maximum priority that a token can have for the adapter to use for transmission. (get-replace)

EarlyTokenReleaseFlag

Specifies whether Early Token Release is used by this station (get). (This attribute has been added in anticipation of the 802.5 standards direction).

AuthorizedFunctionClasses

specifies the function classes that a station is enabled to transmit (get-replace.

RingStationStatus

specifies the current state of the sending ring station's microcode. Its contents are implementation dependent (get).

PhysicalDropNumber

specifies the assigned physical location of the sending ring station. This value is available only if manually assigned (get-replace).

SoftErrorReportTimer

specifies the time-out value for the ring station's T(soft-error-report) timer. (get-replace)

- **Ethernet Layer 1**

EthernetLayer1Name

specifies the naming attributes for this managed object class (get).

SegmentDataRate

specifies the data rate of the segment on which the MAC resides.

IEEEVenderCode

identifies the manufacture of this adapter (get). It is assigned by IEEE and for those without an IEEE vendor code, a value of 0xFFFFFF is used instead.

VendorAdapterCode

identifies the model/type of this adapter (get).

AdapterNumber

specifies the adapter number for this particular adapter out of all the adapters in the box (get).

VendorAdapterDescription

is a string containing a description of the adapter (get).

- **Ethernet Layer 2 MAC**

IndMACAdress

specifies the address of the device on the segment (get).

UniversallyAdministeredAddress

specifies the default address shipped with the device (get).

MulticastAddress

specifies the multicast address that the device has opened (get-replace).

MACDriverVersionNumber

specifies the version number of the MAC driver (get).

MACType

specifies the type of MAC protocol header that the MAC driver supports (get).

AdapterInterruptLevel

specifies the level of interrupt that this adapter uses (get).

SegmentNumber

specifies the segment number on which the MAC resides (get-replace).

FilterStatus

specifies the current packet filter setting (get).

- **LAN LAN Layer 2 - LLC**

LLCName

specifies the naming attribute for this object class.

resource Type ID
specifies the manufacturer of the station and the IEEE 802 standards which apply to the resource. (get)

activeLSAPs
specifies the set of all currently active LSAPs. (get)

maximum LSAPs Configured
specifies the maximum number of SAPs which this station can support (get)

services Supported
specifies the LLC types of service which this station will support. (get)

Number of Active LSAPs
specifies the number of active LSAPs for this LLC Entity (get)

Maximum Link Stations Configured
specifies the number of link stations that can be configured at this LLC entity (get)

Number of Active Link Stations
specifies the number of active link stations at this LLC Entity (get)

Number of Available Link Stations
specifies the number of link stations that can be opened for link connectiors. This is the maximum number of link stations minus the number of active stations minus the number of link stations in disconnect mode. (get)

- **LSAP**

LSAPid
specifies the unique ID of this LSAP (0-127) (get)
supported Types
specifies the type or types of service supported by this LSAP. (get)

activated Types
specifies the type or types of service currently activated at this LSAP. (get)

maximum LLC Information Field Size
specifies the maximum length SDU that the LSAP will accommodate. (get)

active Connections
specifies a set of LLC Remote Address values, one for each active connection at this LSAP. (get)

maximum Link Stations Configured
specifies the maximum number of link station that can be opened at this LSAP (get)

maximumPDUN3
specifies the maximum size of a Type 3 command PDU, if Type 3 is supported (get-replace)

acknowledgementTime
specifies the time interval during which the LLC expects to receive a response to an acknowledged connectionless request. (Type 3) (get-replace)

type3ReceiveResources
causes the LSAP entity to enable or disable the resources for reception of the data field of Type 3 command PDUs sent to this LSAP. (If a Type 3 PDU

containing data is received while disabled, the UN status code (as defined in LLC Type 3) is retur in the CCCC field of the response PDU.) (get-replace)

- **LSAP Pair**

  LSAPPairName
  specifies the name used to uniquely identify the LSAP Pair (get)

  LSAPPairId
  specifies the link station identifier (LSAP, MAC address, remote MAC address, remote LSAP) (get)

  k
  specifies the value of the send window. (get-replace)

  rW
  specifies the value of the receive window. (get-replace)

  maximumRetransmissions
  specifies the number of times that the LLC type 2 should attempt to realize a successful PDU transfer (get-replace)

  t1Timer
  specifies the timeout value of the reply timer (T1) (get-replace)

  t2Timer
  specifies the timeout value of the receiver acknowledgement timer (get-replace)

  tlTimer
  specifies the timeout value of the inactivity timer (TI) (get-replace)

  maxIField
  specifies the maximum I-field length (get-replace)

  maxOutIncrement
  specifies the dynamic window step (get-replace)

  Access Priority
  specifies the access priority used to transmit PDUs (get-replace)

  route
  specifies the route that this lsap pair uses to communicate with the remote station. It exists only if the lsap pair has knowledge of this information. (get)

- **Resource Management**

  primary name
  specifies the name of the LAN Station manager (must be unique) (get)

  RegisteredList
  specifies what resources this Station Manager manages and the managing processes that have requested events. (get)

  Architecture Release Level
  specifies the architecture level of the implementation (get)

equipment related objects
: specifies a set of object identifiers that represents object classes that model the equipment in which the station manager resides. (get)

- **Name Management**

  NMName
  : the naming attribute for this object class (get) for a heartbeat (get-replace)

  Find Timeout
  : the time a station waits for a response to a FIND before retrying (get-replace)

  Find Retry Limit
  : the number of times a station sends a FIND (get-replace)

  ListOfRegularIdentifiers
  : a list of regular identifiers (and their protocol ids) that have been registered with the station manager (get)

  ListOfGroupIdentifiers
  : a list of group identifiers (and their protocol ids) that have been registered with the station manager (get)

  Discovery Server Flag
  : indicates if a discovery server is present and whether to use centralized or distributed address resolution (get-replace)

  Discovery Server Address
  : specifies the MAC address of the discovery server, if one exists. If not, the value is all zeros. (get-replace)

The following objects are defined for products with special functions:

- CAU
- Attachment Module
- Hubs
- Concentrators

Chapter 21. Templates

21.1 Managed Object Templates

The templates used in this chapter are defined in *SMI Part 4: Guidelines for the Definition of Managed Objects (GDMO)*. They are not reproduced in this book.

21.1.1 EthernetLayer1 MANAGED OBJECT CLASS

```
EthernetLayer1 MANAGED OBJECT CLASS
 DERIVED FROM                    MSOOB-TOP ;

 CHARACTERIZED BY:
  ATTRIBUTES
      EthernetLayer1Name         GET,
      MSOAT-SegmentDataRate      GET,
      IEEEVendorCode             GET,
      VendorAdapterCode          GET,
      MSOAT-AdapterNumber        GET,
      VendorAdapterDescription   GET ;

REGISTERED AS { x } ;
```

21.1.2 EthernetLayer2MAC MANAGED OBJECT CLASS

```
EthernetLayer2MAC MANAGED OBJECT CLASS
 DERIVED FROM                    MSOOB-TOP ;

 CHARACTERIZED BY:
  ATTRIBUTES
      MSOAT-IndMACAddress        GET,
      MSOAT-UniversallyAdministeredAddress GET,
      MSOAT-MulticastAddress     GET
      MSOAT-MACDriverVersionNumber GET,
      MSOAT-MACType              GET,
      MSOAT-AdapterInterruptLevel GET,
      MSOAT-SegmentNumber        GET,
      MSOAT-FilterStatus         GET,
  NOTIFICATIONS
      MSONT-GeneralReport ;

REGISTERED AS { x } ;
```

21.1.3 MSOOB-AttachmentModule MANAGED OBJECT CLASS

```
MSOOB-AttachmentModule  MANAGED OBJECT CLASS
 DERIVED FROM                    MSOOB-TOP ;
 CHARACTERIZED BY:
  BEHAVIOUR DEFINITIONS
      MSOBV-AttachmentModule;
      MSOBV-AMAlarms;
  ATTRIBUTES
      MSOAT-AMName               GET,
      MSOAT-TopologyInfo         GET,
```

```
        MSOAT-AMStatus              REPLACE,
        MSOAT-LobeStatus            REPLACE,
    NOTIFICATIONS
        MSONT-GeneralReport ;

REGISTERED AS { 1 3 12 2 1004 213 } ;
```

21.1.4 MSOOB-Counter MANAGED OBJECT CLASS

```
MSOOB-Counter MANAGED OBJECT CLASS
 DERIVED FROM                   MSOOB-Top;
 CHARACTERIZED BY:
  BEHAVIOUR DEFINITIONS         MSOBV-Counter;
  ATTRIBUTES
      MSOAT-CounterName         GET,
      MSOAT-CurrentCountValue   GET,
      MSOAT-CounterMaxValue     GET;

REGISTERED AS { x } ;
```

21.1.5 MSOOB-Environment MANAGED OBJECT CLASS

```
MSOOB-Environment   MANAGED OBJECT CLASS
 DERIVED FROM                   MSOOB-TOP ;
 CHARACTERIZED BY:
  BEHAVIUOR DEFINITIONS
      MSOBV-Environment;
  ATTRIBUTES
      MSOAT-EnvName             GET,
      MSOAT-Location            GET-REPLACE,
      MSOAT-MachineType         GET,
      MSOAT-SerialNumber        GET,
      MSOAT-UserDefinedData     GET-REPLACE ;

REGISTERED AS { 1 3 12 2 1004 524 } ;
```

21.1.6 MSOOB-IAU MANAGED OBJECT CLASS

```
MSOOB-IAU   MANAGED OBJECT CLASS
 DERIVED FROM                   MSOOB-TOP ;
 CHARACTERIZED BY:
  BEHAVIOUR DEFINITIONS
      MSOBV-IAU,
      MSOBV-IAUAlarms;
  ATTRIBUTES
      MSOAT-AccessUnitName      GET,
      MSOAT-IAUVitalInfo        GET,
      MSOAT-ReconParameters     GET-REPLACE,
      MSOAT-IAUStatus           GET,
      MSOAT-Password            REPLACE ;
  OPERATIONS
   ACTIONS
      MSOAC-SoftReset,
      MSOAC-RPUenable,
      MSOAC-IAUWrap ;
  NOTIFICATIONS
      MSONT-CommunicationAlarmConfirmed,
```

```
        MSONT-EnvironmentalAlarmConfirmed,
        MSONT-GeneralReport ;

    REGISTERED AS { 1 3 12 2 1004 212 } ;
```

21.1.7 MSOOB-LANLayer2LLC MANAGED OBJECT CLASS

```
MSOOB-LANLayer2LLC   MANAGED OBJECT CLASS
 DERIVED FROM                    MSOOB-LayerWithCounters ;
 CHARACTERIZED BY:
  BEHAVIOUR DEFINITIONS
      MSOBV-TokenRingLayer2LLC;
  ATTRIBUTES
      MSOAT-LLCName              GET,
      MSOAT-ResourceTypeID       GET,
      MSOAT-ActiveLSAPs          GET,
      MSOAT-MaximumLSAPsConfigured GET,
      MSOAT-NumberOfActiveLSAPs GET,
      MSOAT-MaximumLinkStationsConfigured GET,
      MSOAT-NumberOfActiveLinkStations GET,
      MSOAT-NumberOfAvailableLinkStations GET,
      MSOAT-ServicesSupported    GET ;
  OPERATIONS
   ACTIONS
      MSOAC-Reinitialize,
  NOTIFICATIONS
      MSONT-GeneralReport ;

REGISTERED AS { 1 3 12 2 1004 525 } ;
```

21.1.8 MSOOB-LayerWithCounters MANAGED OBJECT CLASS

```
MSOOB-LayerWithCounters MANAGED OBJECT CLASS
 DERIVED FROM                    MSOOB-Top;
 CHARACTERIZED BY:
  BEHAVIOUR DEFINITIONS          MSOBV-LayerWithCounters;
  OPERATIONS
    ACTIONS
        MSOAC-GetCounters;

REGISTERED AS { x } ;
```

21.1.9 MSOOB-LibraryObject MANAGED OBJECT CLASS

```
MSOOB-LibraryObject MANAGED OBJECT CLASS
 DERIVED FROM                    MSOOB-Top;
 CHARACTERIZED BY:
  BEHAVIOUR DEFINITIONS          MSOBV-LibraryObject;
  ATTRIBUTES
      MSOAT-LibraryObjectName    GET
      REQUIRED VALUES SUPPORT-OBJECTS.LibraryObjectNameValue;

REGISTERED AS { x } ;
```

21.1.10 MSOOB-LSAPEntity MANAGED OBJECT CLASS

```
MSOOB-LSAPEntity   MANAGED OBJECT CLASS
 DERIVED FROM                          MSOOB-LayerWithCounters ;
 CHARACTERIZED BY:
  BEHAVIOUR DEFINITIONS
      MSOBV-LSAPEntity;
  ATTRIBUTES
      MSOAT-LSAPid                     GET,
      MSOAT-SupportedTypes             GET,
      MSOAT-ActivatedTypes             GET,
      MSOAT-MaximumLLCInformationFieldSize GET,
      MSOAT-ActiveConnections          GET,
      MSOAT-MaximumLinkStationsConfigured GET ;
 OPERATIONS
  ACTIONS
      MSOAC-ActivateSAP,
      MSOAC-DeactivateSAP ;
 NOTIFICATIONS
      MSONT-GeneralReport ;
 PACKAGE   MSOCP-LSAP
      PRESENT IF Type3 LLC is supported

REGISTERED AS { 1 3 12 2 1004 526 } ;
```

21.1.11 MSOOB-LSAPPairEntity MANAGED OBJECT CLASS

```
MSOOB-LSAPPairEntity   MANAGED OBJECT CLASS
 DERIVED FROM                          MSOOB-LayerWithCounters ;
 CHARACTERIZED BY:
  BEHAVIOUR DEFINITIONS
      MSOBV-LSAPPairEntity,
      MSOBV-LSAPPairEntityAlarm;
  ATTRIBUTES
      MSOAT-LSAPPairName               GET,
      MSOAT-LSAPPairId                 GET,
      MSOAT-K                          GET-REPLACE,
      MSOAT-RW                         GET-REPLACE,
      MSOAT-MaximumRetransmissions GET-REPLACE,
      MSOAT-T1Timer                    GET-REPLACE,
      MSOAT-T2Timer                    GET-REPLACE,
      MSOAT-TiTimer                    GET-REPLACE,
      MSOAT-MaxOutIncrement            GET-REPLACE,
      MSOAT-MaxLLCInformationFieldSize GET-REPLACE,
      MSOAT-AccessPriority             GET-REPLACE ;
 OPERATIONS
  ACTIONS
      MSOAC-CorrelatorExchange
 NOTIFICATIONS
      MSONT-CommunicationAlarmUnConfirmed,
      MSONT-CommunicationAlarmConfirmed,
      MSONT-GeneralReport ;
 PACKAGE   MSOCP-LSAPPairRoute
      PRESENT IF Route known to the LSAP Pair object

REGISTERED AS { 1 3 12 2 1004 527 } ;
```

21.1.12 MSOOB-ResourceManagement MANAGED OBJECT CLASS

```
MSOOB-ResourceManagement    MANAGED OBJECT CLASS
 DERIVED FROM                    MSOOB-TOP ;
 CHARACTERIZED BY:
  BEHAVIOUR DEFINITIONS
      MSOBV-ResourceManagement;
  ATTRIBUTES
      MSOAT-PrimaryName          GET,
      MSOAT-RegisteredList       GET,
      MSOAT-ArchReleaseLevel     GET,
      MSOAT-EquipmentRelatedObjects GET ;
  OPERATIONS
   ACTIONS
      MSOAC-RegisterRequest,
      MSOAC-DeregisterRequest,
      MSOAC-MultipleFunctionRegisterRequest,
      MSOAC-MultipleFunctionDeregisterRequest,
      MSOAC-RemoveStation,
      MSOAC-RegisterCheck ;
  NOTIFICATIONS
      MSONT-FunctionPresent,
      MSONT-Deregister ;
      MSONT-MultipleFunctionPresent,
      MSONT-MultipleFunctionDeregister ;

REGISTERED AS { 1 3 12 2 1004 215 } ;
```

21.1.13 MSOOB-ThresholdControlInitialValues MANAGED OBJECT CLASS

```
MSOOB-ThresholdControlInitialValues MANAGED OBJECT CLASS
 DERIVED FROM                    MSOOB-Top;
 CHARACTERIZED BY:
  BEHAVIOUR DEFINITIONS          MSOBV-ThresholdControlInitialValues;
  ATTRIBUTES
      MSOAT-ThresholdControlInitialValuesName GET,
      MSOAT-MaxSampleInterval    GET-REPLACE
      REQUIRED VALUES LAYER-COUNTERS.RequiredMaxSampleInterval,
      MSOAT-TriggerList          GET-REPLACE
                                 ADD-REMOVE
                                 SET TO DEFAULT
                                 DEFAULT VALUE {},
      MSOAT-PolicyTargetScope    GET;

REGISTERED AS { x } ;
```

21.1.14 MSOOB-ThresholdControl MANAGED OBJECT CLASS

```
MSOOB-ThresholdControl MANAGED OBJECT CLASS
 DERIVED FROM                    MSOOB-ManagerPointer;
 CHARACTERIZED BY:
  BEHAVIOUR DEFINITIONS          MSOBV-ThresholdControl;
  ATTRIBUTES
      MSOAT-ThresholdControlName GET,
      MSOAT-ThresholdPolicyPointer GET
                                 DEFAULT VALUE NULL,
      MSOAT-ReportSwitch         GET-REPLACE,
```

```
        MSOAT-MaxSampleInterval   GET-REPLACE
        REQUIRED VALUES LAYER-COUNTERS.RequiredMaxSampleInterval,
        MSOAT-TriggerList         GET-REPLACE
                                  ADD-REMOVE
                                  SET TO DEFAULT
                                  DEFAULT VALUE {};
    OPERATIONS
      CREATE  with-automatic-instance-naming;
      DELETE;
    NOTIFICATIONS
        MSONT-CounterSetReport;

  REGISTERED AS { x } ;
```

21.1.15 MSOOB-TokenRingLayer1 MANAGED OBJECT CLASS

```
  MSOOB-TokenRingLayer1   MANAGED OBJECT CLASS
    DERIVED FROM                  MSOOB-TOP ;
    CHARACTERIZED BY:
     BEHAVIOUR DEFINITIONS
        MSOBV-TokenRingLayer1;
     ATTRIBUTES
        MSOAT-TRLayer1Name        GET,
        MSOAT-AccessUnitId        GET-REPLACE,
        MSOAT-SegmentDataRate     GET,
        MSOAT-LobeReceptacleNumber GET-REPLACE,
        MSOAT-AMNumber            GET-REPLACE,
        MSOAT-WallFacePlateNumber GET-REPLACE,
        MSOAT-AdapterNumber       GET ;
    PACKAGE  MSOCP-RingUtilization
        PRESENT IF Adapter can calculate the Ring Utilization ;

  REGISTERED AS { 1 3 12 2 1004 214 } ;
```

21.1.16 MSOOB-TokenRingLayer2MAC MANAGED OBJECT CLASS

```
  MSOOB-TokenRingLayer2MAC   MANAGED OBJECT CLASS
    DERIVED FROM                  MSOOB-TOP ;
    CHARACTERIZED BY:
     BEHAVIOUR DEFINITIONS
        MSOBV-TokenRingLayer2MAC,
        MSOBV-TRMACAlarm;
     ATTRIBUTES
        MSOAT-IndMACAddress       GET,
        MSOAT-UniversallyAdministeredAddress GET,
        MSOAT-FunctionalAddress   GET-REPLACE,
        MSOAT-GroupAddress        GET-REPLACE,
        MSOAT-NAUN                GET,
        MSOAT-SegmentNumber       GET-REPLACE,
        MSOAT-MicrocodeLevel      GET,
        MSOAT-ProductInstanceId   GET,
        MSOAT-AccessPriority      GET-REPLACE,
        MSOAT-SoftErrorReportTimer GET-REPLACE,
        MSOAT-AuthorizedFunctionClasses GET-REPLACE,
        MSOAT-PhysicalDropNumber GET-REPLACE,
        MSOAT-RingStationStatus   GET,
        MSOAT-EarlyTokenReleaseFlag GET ;
```

```
    NOTIFICATIONS
        MSONT-CommunicationAlarmConfirmed ;

REGISTERED AS { 1 3 12 2 1004 220 } ;
```

21.2 Attribute Templates

The templates used in this chapter are defined in *SMI Part 4: Guidelines for the Definition of Managed Objects (GDMO)*. They are not reproduced in this book.

21.2.1 MSOAT-AccessPriority ATTRIBUTE

```
MSOAT-AccessPriority ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.AccessPriority ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 463 } ;
```

21.2.2 MSOAT-AccessUnitId ATTRIBUTE

```
MSOAT-AccessUnitId ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.AccessUnitId ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.3 MSOAT-AccessUnitName ATTRIBUTE

```
MSOAT-AccessUnitName ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.AccessUnitName ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 195 } ;
```

21.2.4 MSOAT-AcknowledgeTimer ATTRIBUTE

```
MSOAT-AcknowledgeTimer ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.AcknowledgementTime ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 464 } ;
```

21.2.5 MSOAT-ActivatedTypes ATTRIBUTE

```
MSOAT-ActivatedTypes ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.ActivatedTypes ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 465 } ;
```

21.2.6 MSOAT-ActiveConnections ATTRIBUTE

```
MSOAT-ActiveConnections ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.ActiveConnections :
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 466 } ;
```

21.2.7 MSOAT-ActiveLSAPs ATTRIBUTE

```
MSOAT-ActiveLSAPs ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.ActiveLSAPs ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 467 } ;
```

21.2.8 MSOAT-AdapterNumber ATTRIBUTE

```
MSOAT-AdapterNumber ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.AdapterNumber ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 468 } ;
```

21.2.9 MSOAT-AdapterInterruptLevel ATTRIBUTE

```
MSOAT-AdapterInterruptLevel ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.AdapterInterruptLevel;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.10 MSOAT-ArchReleaseLevel ATTRIBUTE

```
MSOAT-ArchReleaseLevel ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.ArchReleaseLevel ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 469 } ;
```

21.2.11 MSOAT-AuthorizedFunctionClasses ATTRIBUTE

```
MSOAT-AuthorizedFunctionClasses ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.FunctionClasses ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 470 } ;
```

21.2.12 MSOAT-AMName ATTRIBUTE

```
MSOAT-AMName ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.AMName ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 217 } ;
```

21.2.13 MSOAT-AMNumber ATTRIBUTE

```
MSOAT-AMNumber ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.AttachmentModule ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.14 MSOAT-AMStatus ATTRIBUTE

```
MSOAT-AMStatus ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.AMStatus ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 216 } ;
```

21.2.15 MSOAT-CounterMaxValue ATTRIBUTE

```
MSOAT-CounterMaxValue ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAYER-COUNTERS.CounterValue;
 MATCHES FOR                  Equality, Ordering;

REGISTERED AS { x } ;
```

21.2.16 MSOAT-CounterName ATTRIBUTE

```
MSOAT-CounterName ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAYER-COUNTERS.CounterName;
 MATCHES FOR                  Equality;
 BEHAVIOUR                    MSOBV-CounterName;

REGISTERED AS {1 3 12 2 1004 164 };
```

21.2.17 MSOAT-CurrentCountValue ATTRIBUTE

```
MSOAT-CurrentCountValue ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAYER-COUNTERS.CounterValue;
 MATCHES FOR                  Equality, Ordering;

REGISTERED AS { 1 3 12 2 1004 xxx } ;
```

21.2.18 MSOAT-DiscoveryServerAddress ATTRIBUTE

```
MSOAT-DiscoveryServerAddress  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        MACAddress ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 472 } ;
```

21.2.19 MSOAT-DiscoveryServerFlag ATTRIBUTE

```
MSOAT-DiscoveryServerFlag  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        BOOLEAN ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 471 } ;
```

21.2.20 MSOAT-EarlyTokenReleaseFlag ATTRIBUTE

```
MSOAT-EarlyTokenReleaseFlag  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.EarlyTokenReleaseFlag ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 473 } ;
```

21.2.21 MSOAT-EnvName ATTRIBUTE

```
MSOAT-EnvName    ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.EnvName ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 474 } ;
```

21.2.22 MSOAT-EthernetLayer1Name ATTRIBUTE

```
MSOAT-EthernetLayer1Name    ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.EthernetLayer1Name ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.23 MSOAT-FilterStatus ATTRIBUTE

```
MSOAT-EthernetLayer1Name    ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.FilterStatus ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.24 MSOAT-FindRetryLimit ATTRIBUTE

```
MSOAT-FindRetryLimit ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.Timer ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 476 } ;
```

21.2.25 MSOAT-FindTimeout ATTRIBUTE

```
MSOAT-FindTimeout ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.Timer ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 475 } ;
```

21.2.26 MSOAT-FunctionalAddress ATTRIBUTE

```
MSOAT-FunctionalAddress ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MACAddress ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 477 } ;
```

21.2.27 MSOAT-GroupAddress ATTRIBUTE

```
MSOAT-GroupAddress ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.GroupAddress ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 478 } ;
```

21.2.28 MSOAT-HeartbeatRepetitionPeriod ATTRIBUTE

```
MSOAT-HeartbeatRepetitionPeriod ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.Timer ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 479 } ;
```

21.2.29 MSOAT-HeartbeatTimerDuration ATTRIBUTE

```
MSOAT-HeartbeatTimerDuration ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.Timer ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 480 } ;
```

21.2.30 MSOAT-IAUStatus ATTRIBUTE

```
MSOAT-IAUStatus ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.IAUStatus ;
 MATCHES FOR                    Equality;

REGISTERED AS { x } ;
```

21.2.31 MSOAT-IAUVitalInfo ATTRIBUTE

```
MSOAT-IAUVitalInfo ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.IAUVitalInfo ;
 MATCHES FOR                    Equality;

REGISTERED AS { x } ;
```

21.2.32 MSOAT-IEEEVendorCode ATTRIBUTE

```
MSOAT-IEEEVendor Code ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.IEEEVendorCode ;
 MATCHES FOR                    Equality;

REGISTERED AS { x } ;
```

21.2.33 MSOAT-IndMACAddress ATTRIBUTE

```
MSOAT-IndMACAddress ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.MACAddress ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 218 } ;
```

21.2.34 MSOAT-K ATTRIBUTE

```
MSOAT-K      ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.k ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 481 } ;
```

21.2.35 MSOAT-LibraryObjectName ATTRIBUTE

```
MSOAT-LibraryObjectName ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        SUPPORT-OBJECTS.LibraryObjectName;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.36 MSOAT-ListOfGroupTitles ATTRIBUTE

```
MSOAT-ListOfGroupTitles  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.ListOfGroupTitles ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 483 } ;
```

21.2.37 MSOAT-ListOfRegularTitles ATTRIBUTE

```
MSOAT-ListOfRegularTitles  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.ListOfTitles
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 484 } ;
```

21.2.38 MSOAT-LobeReceptacleNumber ATTRIBUTE

```
MSOAT-LobeReceptacleNumber ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.LobeId ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 201 } ;
```

21.2.39 MSOAT-LobeStatus ATTRIBUTE

```
MSOAT-LobeStatus ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.LobeStatus ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.40 MSOAT-Location ATTRIBUTE

```
MSOAT-Location ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.Location ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 485 } ;
```

21.2.41 MSOAT-LLCName ATTRIBUTE

```
MSOAT-LLCName    ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.LLCName ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 482 } ;
```

21.2.42 MSOAT-LSAPid ATTRIBUTE

```
MSOAT-LobeStatus ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.LobeStatus ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.43 MSOAT-LSAPPairId ATTRIBUTE

```
MSOAT-LSAPPairId ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.LSAPPairId ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 487 } ;
```

21.2.44 MSOAT-LSAPPairName ATTRIBUTE

```
MSOAT-LSAPPairName ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.LSAPPAirName ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 488 } ;
```

21.2.45 MSOAT-MACDriverVersionNumber ATTRIBUTE

```
MSOAT-MACDriverVersionNumber ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MACDriverVersionNumber ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.46 MSOAT-MACType ATTRIBUTE

```
MSOAT-MACType ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MACType ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.47 MSOAT-MachineSpecificObjects ATTRIBUTE

```
MSOAT-EquipmentRelatedObjects ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.EquipmentRelatedObjects ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 489 } ;
```

21.2.48 MSOAT-MachineType ATTRIBUTE

```
MSOAT-MachineType  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MachineType ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 490 } ;
```

21.2.49 MSOAT-MaximumLinkStationsConfigured ATTRIBUTE

```
MSOAT-MaximumLinkStationsConfigured ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MaximumLinkStationsConfigured ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 491 } ;
```

21.2.50 MSOAT-MaximumLLCInformationFieldSize ATTRIBUTE

```
MSOAT-MaximumLLCInformationFieldSize ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MaximumLLCInformationFieldSize ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 492 } ;
```

21.2.51 MSOAT-MaximumLSAPsConfigured ATTRIBUTE

```
MSOAT-MaximumLSAPsConfigured ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MaximumSAPsConfigured ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 495 } ;
```

21.2.52 MSOAT-MaximumPDUN3 ATTRIBUTE

```
MSOAT-MaximumPDUN3 ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MaximumPDUN3 ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 493 } ;
```

21.2.53 MSOAT-MaximumRetransmissions ATTRIBUTE

```
MSOAT-MaximumRetransmissions ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MaximumRetransmissions ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 494 } ;
```

21.2.54 MSOAT-MaxOutIncrement ATTRIBUTE

```
MSOAT-MaxOutIncrement ATTRIBUTE
 SINGLE&US.VALUED
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MaxOutIncrement ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 496 } ;
```

21.2.55 MSOAT-MaxSampleInterval ATTRIBUTE

```
MSOAT-MaxSampleInterval ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAYER-COUNTERS.MaxSampleInterval;
 MATCHES FOR                  Equality, Ordering;
 BEHAVIOUR                    MSOBV-MaxSampleInterval;

REGISTERED AS { x } ;
```

21.2.56 MSOAT-MicrocodeLevel ATTRIBUTE

```
MSOAT-MicrocodeLevel ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MicrocodeLevel ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 497 } ;
```

21.2.57 MSOAT-MulticastAddress ATTRIBUTE

```
MSOAT-MulticastAddress ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MulticastAddress ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.58 MSOAT-NumberOfActiveLinkStations ATTRIBUTE

```
MSOAT-NumberOfActiveLinkStations ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        INTEGER ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 502 } ;
```

21.2.59 MSOAT-NumberOfActiveLSAPS ATTRIBUTE

```
MSOAT-NumberOfActiveLSAPS ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        INTEGER ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 500 } ;
```

21.2.60 MSOAT-NumberOfAvailableLinkStations ATTRIBUTE

```
MSOAT-NumberOfAvailableLinkStations  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        INTEGER ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 501 } ;
```

21.2.61 MSOAT-NAUN ATTRIBUTE

```
MSOAT-NAUN ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MACAddress ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 498 } ;
```

21.2.62 MSOAT-NMName ATTRIBUTE

```
MSOAT-NMName    ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.NMName ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 499 } ;
```

21.2.63 MSOAT-Password ATTRIBUTE

```
MSOAT-Password  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.SecurityAccessField ;
 MATCHES FOR                    Equality;

REGISTERED AS { x } ;
```

21.2.64 MSOAT-PhysicalDropNumber ATTRIBUTE

```
MSOAT-PhysicalDropNumber  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          PhysicalDropNumber ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 503 } ;
```

21.2.65 MSOAT-PrimaryName ATTRIBUTE

```
MSOAT-PrimaryName  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          DAARD.RegularTitle ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 456 } ;
```

21.2.66 MSOAT-ProductInstanceId ATTRIBUTE

```
MSOAT-ProductInstanceId ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.ProductInstanceID ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 504 } ;
```

21.2.67 MSOAT-ReconParameters ATTRIBUTE

```
MSOAT-ReconParameters ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.ReconParameters ;
 MATCHES FOR                    Equality;

REGISTERED AS { x } ;
```

21.2.68 MSOAT-RegisteredList ATTRIBUTE

```
MSOAT-RegisteredList  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.RegisteredList ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 505 } ;
```

21.2.69 MSOAT-ReportSwitch ATTRIBUTE

```
MSOAT-ReportSwitch ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAYER-COUNTERS.ReportSwitch;
 MATCHES FOR                    Equality;

REGISTERED AS { x } ;
```

21.2.70 MSOAT-ResourceTypeId ATTRIBUTE

```
MSOAT-ResourceTypeId ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.ResourceTypeId ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 506 } ;
```

21.2.71 MSOAT-RingStationStatus ATTRIBUTE

```
MSOAT-RingStationStatus  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.RingStationStatus ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 508 } ;
```

21.2.72 MSOAT-RingUtilization ATTRIBUTE

```
MSOAT-RingUtilization  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.RingUtilization ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 507 } ;
```

21.2.73 MSOAT-Route ATTRIBUTE

```
MSOAT-Route  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.Route ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 509 } ;
```

21.2.74 MSOAT-RW ATTRIBUTE

```
MSOAT-RW  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.RW ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 510 } ;
```

21.2.75 MSOAT-SegmentDataRate ATTRIBUTE

```
MSOAT-SegmentDataRate ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.SegmentDataRate ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 511 } ;
```

21.2.76 MSOAT-SegmentNumber ATTRIBUTE

```
MSOAT-SegmentNumber ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.SegmentNumber ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 512 } ;
```

21.2.77 MSOAT-SerialNumber ATTRIBUTE

```
MSOAT-SerialNumber  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.SerialNumber ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 513 } ;
```

21.2.78 MSOAT-ServicesSupported ATTRIBUTE

```
MSOAT-ServicesSupported ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.ServicesSupported ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 514 } ;
```

21.2.79 MSOAT-SoftErrorReportTimer ATTRIBUTE

```
MSOAT-SoftErrorReportTimer ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.Timer ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 515 } ;
```

21.2.80 MSOAT-SupportedTypes ATTRIBUTE

```
MSOAT-SupportedTypes ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.SupportedTypes ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 516 } ;
```

21.2.81 MSOAT-ThresholdControlInitialValuesName ATTRIBUTE

```
MSOAT-ThresholdControlInitialValuesName ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAYER-COUNTERS.ThresholdControlInitialValuesName;
 MATCHES FOR                    Equality;

REGISTERED AS { x } ;
```

21.2.82 MSOAT-ThresholdControlName ATTRIBUTE

```
MSOAT-ThresholdControlName ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAYER-COUNTERS.ThresholdControlName;
 MATCHES FOR                    Equality;

REGISTERED AS { x } ;
```

21.2.83 MSOAT-TiTimer ATTRIBUTE

```
MSOAT-TiTimer ATTRIBUTE
 WITH ATTRIBUTE SYNTAX          LAN-COMMON.Timer ;
 MATCHES FOR                    Equality;

REGISTERED AS { 1 3 12 2 1004 519 } ;
```

21.2.84 MSOAT-TriggerList ATTRIBUTE

```
MSOAT-TriggerList ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAYER-COUNTERS.TriggerList;
 MATCHES FOR                  Set Comparison;

REGISTERED AS { x } ;
```

21.2.85 MSOAT-TopologyInfo ATTRIBUTE

```
MSOAT-TopologyInfo ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.TopologyInfo ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.86 MSOAT-Type3ReceiveResources ATTRIBUTE

```
MSOAT-Type3ReceiveResources ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.Type3ReceiveResources ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 520 } ;
```

21.2.87 MSOAT-TRLayer1Name ATTRIBUTE

```
MSOAT-TRLayer1Name ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.TRLayer1Name ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.88 MSOAT-T1Timer ATTRIBUTE

```
MSOAT-T1Timer ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.Timer ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 517 } ;
```

21.2.89 MSOAT-T2Timer ATTRIBUTE

```
MSOAT-T2Timer  ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.Timer ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 518 } ;
```

21.2.90 MSOAT-UniversallyAdministeredAddress ATTRIBUTE

```
MSOAT-UniversallyAdministeredAddress ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.MACAddress ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 521 } ;
```

21.2.91 MSOAT-UserDefinedData ATTRIBUTE

```
MSOAT-UserDefinedData ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.UserData ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 522 } ;
```

21.2.92 MSOAT-VendorAdapterCode ATTRIBUTE

```
MSOAT-VendorAdapterCode ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.VendorAdapterCode ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.93 MSOAT-VendorAdapterDescription ATTRIBUTE

```
MSOAT-VendorAdapterDescription ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.VendorAdapterDescription ;
 MATCHES FOR                  Equality;

REGISTERED AS { x } ;
```

21.2.94 MSOAT-WallFacePlateLabel ATTRIBUTE

```
MSOAT-UserDefinedData ATTRIBUTE
 WITH ATTRIBUTE SYNTAX        LAN-COMMON.UserData ;
 MATCHES FOR                  Equality;

REGISTERED AS { 1 3 12 2 1004 523 } ;
```

21.3 Conditional Package Templates

The templates used in this chapter are defined in *SMI Part 4: Guidelines for the Definition of Managed Objects (GDMO)*. They are not reproduced in this book.

21.3.1 MSOCP-LSAPPairRoute CONDITIONAL PACKAGE

MSOCP-LSAPPairRoute CONDITIONAL PACKAGE

BEHAVIOUR DEFINITIONS MSOBV-LSAPPairRoute ;

ATTRIBUTES
MSOAT-Route GET,

REGISTERED AS { x } ;

21.3.2 MSOCP-LSAPType3 CONDITIONAL PACKAGE

MSOCP-LSAPType3 CONDITIONAL PACKAGE

BEHAVIOUR DEFINITIONS MSOBV-LSAPType3Behaviour ;

ATTRIBUTES
MSOAT-MaximumPDUN3 GET-REPLACE,
MSOAT-AcknowledgeTimer GET-REPLACE,
MSOAT-Type3ReceiveResources GET-REPLACE ;

REGISTERED AS { x } ;

21.3.3 MSOCP-RingUtilization CONDITIONAL PACKAGE

MSOCP-RingUtilization CONDITIONAL PACKAGE

BEHAVIOUR DEFINITIONS MSOBV-RingUtilization ;

ATTRIBUTES
MSOAT-RingUtilization GET ;

REGISTERED AS { x } ;

21.4 Action Templates

21.4.1 MSOAC-ActivateSAP ACTION

```
MSOAC-ActivateSAP  ACTION
  ACTION BEHAVIOUR              MSOBV-ActivateSAP ;
  WITH DATA SYNTAX              LAN-ACTIONS.SAPData ;
  WITH RESULT SYNTAX            LAN-ACTIONS.SAPData ;

REGISTERED AS { 1 3 12 2 1004 459 } ;
```

21.4.2 MSOAC-CorrelatorExchange ACTION

```
MSOAC-CorrelatorExchange  ACTION
  ACTION BEHAVIOUR              MSOBV-CorrelatorExchange ;
  WITH DATA SYNTAX              LAN-ACTIONS.CorrelatorData ;
  WITH RESULT SYNTAX            LAN-ACTIONS.CorrelatorRspData ;

REGISTERED AS { 1 3 12 2 1004 461 } ;
```

21.4.3 MSOAC-DeactivateSAP ACTION

```
MSOAC-DeactivateSAP  ACTION
  ACTION BEHAVIOUR              MSOBV-DeactivateSAP ;
  WITH DATA SYNTAX              LAN-ACTIONS.SAPData ;
  WITH RESULT SYNTAX            LAN-ACTIONS.SAPData ;

REGISTERED AS { 1 3 12 2 1004 460 } ;
```

21.4.4 MSOAC-DeregisterRequest ACTION

```
MSOAC-DeregisterRequest ACTION
  ACTION BEHAVIOUR              MSOBV-DeregisterRequestBehavior
  WITH DATA SYNTAX              LAN-ACTIONS.DeregisterReqData ;

REGISTERED AS { 1 3 12 2 1004 190 } ;
```

21.4.5 MSOAC-GetCounters ACTION

```
MSOAC-GetCounters ACTION
  ACTION BEHAVIOUR              MSOBV-GetCounters;
   WITH RESULT SYNTAX           LAYER-COUNTERS.CounterSetReply;

REGISTERED AS { x } ;
```

21.4.6 MSOAC-IAUWrap ACTION

```
MSOAC-IAUWrap ACTION
  ACTION BEHAVIOUR              IAUWrap ;
  WITH DATA SYNTAX              LAN-ACTIONS.WrapData ;
  WITH RESULT ;

REGISTERED AS { 1 3 12 2 1004 193 } ;
```

21.4.7 MSOAC-MultipleDeregisterRequest ACTION

```
MSOAC-MultipleDeregisterRequest ACTION
 ACTION BEHAVIOUR              MSOBV-MultipleDeregisterRequestBehavior
 WITH DATA SYNTAX              LAN-ACTIONS.MultipleDeregisterReqData ;

REGISTERED AS { x } ;
```

21.4.8 MSOAC-MultipleRegisterRequest ACTION

```
MSOAC-MultipleRegisterRequest ACTION
 ACTION BEHAVIOUR              MSOBV-MultipleRegisterRequestBehaviour ;
 WITH DATA SYNTAX              LAN-ACTIONS.MultipleRegisterReqData ;
 WITH RESULT SYNTAX            LAN-ACTIONS.MultipleRegisterReqRspData ;

REGISTERED AS { x } ;
```

21.4.9 MSOAC-RegisterCheck ACTION

```
MSOAC-RegisterCheck ACTION
 ACTION BEHAVIOUR              MSOBV-RegisterCheckBehavior ;
 WITH DATA SYNTAX              LAN-ACTIONS.RegisterCheckData ;
 WITH RESULT SYNTAX            LAN-ACTIONS.RegisterCheckRspData ;

REGISTERED AS { 1 3 12 2 1004 191 } ;
```

21.4.10 MSOAC-RegisterRequest ACTION

```
MSOAC-RegisterRequest ACTION
 ACTION BEHAVIOUR              MSOBV-RegisterRequestBehaviour ;
 WITH DATA SYNTAX              LAN-ACTIONS.RegisterReqData ;
 WITH RESULT SYNTAX            LAN-ACTIONS.RegisterReqRspData ;

REGISTERED AS { 1 3 12 2 1004 189 } ;
```

21.4.11 MSOAC-Reinitialize ACTION

```
MSOAC-Reinitialize ACTION
 ACTION BEHAVIOUR              MSOBV-Reinitialize ;
 WITH DATA SYNTAX              LAN-ACTIONS.ReinitializeData ;
 WITH RESULT ;

REGISTERED AS { 1 3 12 2 1004 458 } ;
```

21.4.12 MSOAC-RemoveStation ACTION

```
MSOAC-RemoveStation ACTION
 ACTION BEHAVIOUR              MSOBV-RemoveStationBehavior ;
 WITH DATA SYNTAX              LAN-ACTIONS.RemoveStationData ;
 WITH RESULT SYNTAX            LAN-ACTIONS.RemoveStationRspData ;

REGISTERED AS { 1 3 12 2 1004 462 } ;
```

21.4.13 MSOAC-RPUEnable ACTION

```
MSOAC-RPUEnable ACTION
 ACTION BEHAVIOUR          MSOBV-RPUEnable ;
 WITH DATA SYNTAX          LAN-ACTIONS.RPUEnableData ;

REGISTERED AS { 1 3 12 2 1004 194 } ;
```

21.4.14 MSOAC-SoftReset ACTION

```
MSOAC-SoftReset ACTION
 ACTION BEHAVIOUR          MSOBV-SoftResetData ;
 WITH DATA SYNTAX          LAN-ACTIONS.SoftResetData ;

REGISTERED AS { 1 3 12 2 1004 192 } ;
```

21.5 Notification Templates

21.5.1 MSONT-CounterSetReport NOTIFICATION

```
MSONT-CounterSetReport NOTIFICATION
 BEHAVIOUR                    MSOBV-CounterSetReport
 WITH DATA SYNTAX             LAYER-COUNTERS.CounterSetReport;
 WITH RESULT ;
REGISTERED AS { x } ;
```

21.5.2 MSONT-FunctionPresent NOTIFICATION

```
MSONT-FunctionPresent  NOTIFICATION
 BEHAVIOUR                    MSOBV-FunctionPresent ;
 MODE       NON-CONFIRMED ;
 WITH INFORMATION SYNTAX      LAN-NOTIFIES.FunctionPresent ;

REGISTERED AS { 1 3 12 2 1004 208 } ;
```

21.5.3 MSONT-Deregister NOTIFICATION

```
MSONT-Deregister NOTIFICATION
 BEHAVIOUR                    MSOBV-Deregister ;
 MODE       NON-CONFIRMED
 WITH INFORMATION SYNTAX      LAN-NOTIFIES.Deregister ;

REGISTERED AS { 1 3 12 2 1004 209 } ;
```

21.5.4 MSONT-MultipleFunctionDeregister NOTIFICATION

```
MSOBV-MultipleFunctionDeregister NOTIFICATION
 BEHAVIOUR                    MSOBV-MultipleFunctionDeregister ;
 MODE       NON-CONFIRMED ;
 WITH DATA SYNTAX             LAN-NOTIFIES.Deregister ;

REGISTERED AS { x } ;
```

21.5.5 MSONT-MultipleFunctionPresent NOTIFICATION

```
MSONT-MultipleFunctionPresent  NOTIFICATION
 BEHAVIOUR                    MSOBV-MultipleFunctionPresent ;
 MODE       NON-CONFIRMED ;
 WITH INFORMATION SYNTAX      LAN-NOTIFIES.MultipleFunctionPresent ;

REGISTERED AS { x } ;
```

21.5.6 MSONT-GeneralReport NOTIFICATION

```
MSONT-GeneralReport NOTIFICATION
 BEHAVIOUR                    MSOBV-GenRepBehaviour ;
 MODE       CONFIRMED AND NON-CONFIRMED ;
 WITH DATA SYNTAX             LAN-NOTIFIES.GeneralReport ;
REGISTERED AS { 1 3 12 2 1004 211 } ;
```

21.6 Name Bindings Templates

21.6.1 MSONB-ETHERPHY NAME BINDING

```
MSONB-ETHERBY NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-EthernetLayer1 ;
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-EthernetLayer2MAC ;
 ATTRIBUTE                   EthernetLayerName ;
REGISTERED AS { x } ;
```

21.6.2 MSONB-LSAP NAME BINDING

```
MSONB-LSAP   NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-LSAPEntity ;
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-TokenRingLayer2MAC ;
 ATTRIBUTE                   MSOAT-LSAPId ;
REGISTERED AS { x } ;
```

21.6.3 MSONB-LLC NAME BINDING

```
MSONB-LLC  NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-LANLayer2LLC ;
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-TokenRingLayer2MAC ;
 ATTRIBUTE                   LLCName ;
REGISTERED AS { x } ;
```

21.6.4 MSONB-ThresholdControlInitialValues NAME BINDING

```
MSONB-ThresholdControlInitialValues NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-ThresholdControlInitialValues;
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-LibraryObject;
 ATTRIBUTE                   MSOAT-ThresholdControlInitialValuesName;
REGISTERED AS { x } ;
```

21.6.5 MSONB-TRPHY NAME BINDING

```
MSONB-TRPHY  NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-TokenRingLayer1
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-TokenRingLayer2MAC
 ATTRIBUTE                   TRLayer1Name ;
REGISTERED AS { x } ;
```

21.6.6 MSONB-ENVIRONMENT NAME BINDING

```
MSONB-ENVIRONMENT  NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-Environment ;
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-ResourceManagement ;
 ATTRIBUTE                   EnvName ;
REGISTERED AS { x } ;
```

21.6.7 MSONB-LLCCounter NAME BINDING

```
MSONB-LLCCounter NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-Counter ;
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-LANLayer2LLC ;
 ATTRIBUTE                   MSOAT-CounterName ;

REGISTERED AS { x } ;
```

21.6.8 MSONB-LLCThresholdControl NAME BINDING

```
MSONB-LLCThresholdControl NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-ThresholdControl ;
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-LANLayer2LLC ;
 ATTRIBUTE                   MSOAT-ThresholdControlName ;

REGISTERED AS { x } ;
```

21.6.9 MSONB-LSAPCOUNTER NAME BINDING

```
MSONB-LSAPCOUNTER NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-Counter ;
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-LSAPEntity ;
 ATTRIBUTE                   MSOAT-CounterName ;

REGISTERED AS { x } ;
```

21.6.10 MSONB-LSAPPAIR NAME BINDING

```
MSONB-LSAPPAIR NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-LSAPPairEntity ;
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-LSAPEntity ;
 ATTRIBUTE                   MSOAT-LSAPPairName ;
REGISTERED AS { x } ;
```

21.6.11 MSONB-LSAPPairCounter NAME BINDING

```
MSONB-LSAPPairCounter NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-Counter ;
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-LSAPPairEntity ;
 ATTRIBUTE                   MSOAT-CounterName ;
REGISTERED AS { x } ;
```

21.6.12 MSONB-LSAPPairThresholdControl NAME BINDING

```
MSONB-LSAPPairThresholdControl NAME BINDING
 SUBORDINATE OBJECT CLASS    MSOOB-ThresholdControl ;
 NAMED BY
 SUPERIOR OBJECT CLASS       MSOOB-LSAPPairEntity ;
 ATTRIBUTE                   MSOAT-ThresholdControlName ;

REGISTERED AS { x } ;
```

21.6.13 MSONB-LSAPTHRESHOLDCONTROL NAME BINDING

```
MSONB-LSAPTHRESHOLDCONTROL  NAME BINDING
  SUBORDINATE OBJECT CLASS    MSOOB-ThresholdControl ;
  NAMED BY
  SUPERIOR OBJECT CLASS       MSOOB-LSAPEntity ;
  ATTRIBUTE                   MSOAT-ThresholdControlName ;
REGISTERED AS { x } ;
```

21.7 Behaviours Templates

21.7.1 MSOBV-ActivateSAP BEHAVIOUR

MSOBV-ActivateSAP BEHAVIOUR
DEFINED AS

This CONFIRMED action is to cause the LSAP to be activated for Type 1, Type 2, or Type 3 operation or any combination of those types.

;

21.7.2 MSOBV-CounterName BEHAVIOUR

MSOBV-CounterName BEHAVIOUR
DEFINED AS

Defined values are listed in 21.8.1, "Values of the MSOAT-CounterName ATTRIBUTE" on page 21-43

;

21.7.3 MSOBV-CounterSetReport BEHAVIOUR

MSOBV-CounterSetReport BEHAVIOUR
DEFINED AS

The MSONT-CounterSetReport is emitted by a managed object as a result of a number of different occurrences. The following defines which elements are to be included in the notification for each of the cases in which it is emitted:

Deletion of this managed object (tag = 0):

- counterSetData

Deletion of the containing managed object (tag = 1):

- counterSetData

Wrap of a counter in the counter set (tag = 2):

- counterSetData
- wrapData

See 21.7.18, ".MSOBV-LayerWithCounters BEHAVIOUR" on page 21-34 for a description of what comprises a counter set.

Satisfaction of a threshold trigger condition (tag = 3):

- counterSetData
- numeratorData
- denominatorData (if the trigger has a denominator)

See 21.7.33, "MSOBV-ThresholdControl BEHAVIOUR" on page 21-38 for a description of how a threshold trigger operates ;

21.7.4 MSOBV-GenRepBehaviour BEHAVIOUR

MSOBV-GenRepBehaviour BEHAVIOUR
DEFINED AS
This event may be either confirmed or unconfirmed. This notification is sent under the following conditions:

Link Station Connected
Link Station Disconnected
New Communication Address for the IAU
IAU Lobe Status Change
IAU Attachment Module Status Change
Set Occurred
Non-Registered Managing Process Performed a GET
Device Online
Device Offline ;

21.7.5 MSOBV-Counter BEHAVIOUR

MSOBV-Counter BEHAVIOUR
DEFINED AS

A counter object's sole role is to count: all threshold-related activity takes place in the MSOOB-ThresholdControl objects. The behaviour of a counter object x is described in terms of the following parameters:

c(x)	current value of the counter. This is the value of the MSOAT-CurrentCountValue attribute.
i(x)	amount by which the counter is being incremented. This value is always positive, and it never exceeds maxValue(x). It is not represented by an attribute of MSOOB-Counter.
maxValue(x)	the maximum value of the counter. This is the value of the MSOAT-CounterMaxValue attribute.

MaxValue(x) here is just the size of the counter, rather than an independently-specified value.

```
** At creation of the counter object:

   c(x) := 0
   ** This is the only time the counter is set to 0


** When the counter is incremented:

   If c(x) + i(x) <= maxValue(x)
      then
        ** If the counter does not wrap, simply increment it.
        c(x) := c(x) + i(x)
      else
        ** If a wrap occurs, the count goes to the proper
        ** offset from 0.  Other objects, for example, MSOOB-ThresholdControl,
        ** can monitor c(x) and note that it has decreased.
        c(x) := c(x) - (maxValue(x) + 1) + i(x)  ;
```

21.7.6 MSOBV-DeactivateSAP BEHAVIOUR

MSOBV-DeactivateSAP BEHAVIOUR
DEFINED AS
This CONFIRMED action is to cause the LSAP to be deactivated for Type 1, Type 2, or Type 3 operation or any combination of those types ;

21.7.7 MSOBV-Deregister BEHAVIOUR

MSOBV-Deregister BEHAVIOUR
DEFINED AS
This is an unconfirmed event report to indicate that a function no longer requires management from the managing processs and has removed the managing process from the registered list ;

21.7.8 MSOBV-Environment BEHAVIOUR

MSOBV-Environment BEHAVIOUR
DEFINED AS
This object class defines the configuration of the box in which the station manager resides.
Its attributes include location, machine type, serial number and pointers to objects that further describe the makeup of the box in which the station manager resides. It has no actions or notifications ;

21.7.9 MSOBV-EthernetGeneralReport BEHAVIOUR

MSOBV-EthernetGeneralReport BEHAVIOUR
DEFINED AS
This event is defined as unconfirmed. This notification is send under the following conditions:

Frame received with overrun error
Late collision(out-of-window) error
Carrier lost during transmission
Frame transmitted with CD heartbeat error
Frame with underrun error ;

21.7.10 MSOBV-EthernetLayer1 BEHAVIOUR

MSOBV-EthernetLayer1 BEHAVIOUR
DEFINED AS
This object class defines the physical information pertaining to an Ethernet adapter. Operation of the Ethernet layer 1 is defined in the ANSI/IEEE 802.3 and ISO/DIS 8802/3 Standard ;

21.7.11 MSOBV-EthernetLayer2MAC BEHAVIOUR

MSOBV-EthernetLayer2MAC BEHAVIOUR
DEFINED AS

This object class defines the physical information pertaining to an Ethernet adapter. Operation of the Ethernet layer 1 is defined in the ANSI/IEEE 802.3 and ISO/DIS 8802/3 Standard ;

21.7.12 MSOBV-FunctionPresent BEHAVIOUR

MSOBV-FunctionPresent BEHAVIOUR
DEFINED AS

This is an unconfirmed event report to indicate that a function has come on-line and is looking for registration ;

21.7.13 MSOBV-GetCounters BEHAVIOUR

MSOBV-GetCounters BEHAVIOUR
DEFINED AS

A managed object returns, in response to this action, the names and current values of all counters contained immediately below it in the containment hierarchy. To allow for extendibility, a managed object does not do this by retrieving values from a fixed set of counters. Rather, it performs the equivalent of a scoped GET beneath itself, retrieving data from all counters that it finds there. As would be the case with an actual GET, the counters themselves are unaffected by this action. Any GET error results that may be returned are not included in the result of this action ;

21.7.14 MSOBV-IAU BEHAVIOUR

MSOBV-IAU BEHAVIOUR
DEFINED AS

This object class provides management of an intelligent access unit. It provides information on topology and status.

The password attribute is used to limit sets and actions directed at this object class. The access control field in the CMIP frames is checked for a match with the password before the set or action is performed. The password is required in action and set frames ;

21.7.15 MSOBV-IAUAlarms BEHAVIOUR

MSOBV-IAUAlarms BEHAVIOUR
DEFINED AS

Attributes of interest in Alarms:
- All IAU attributes may be of interest in Alarms.

Alarm conditions:
- A status change for the IAU back-up path.
- A wrap status change for an IAU.
- An IAU base unit internal error.

- A mismatch between the number of lobes active and addresses known in an IAU.
- An IAU ignoring a Force Remove command.
- Removal of a lobe or attachment module by an IAU ;

21.7.16 MSOBV-IAUWrap BEHAVIOUR

MSOBV-IAUWrap BEHAVIOUR
DEFINED AS

This CONFIRMED action is to cause the IAU to perform a wrap ;

21.7.17 MSOBV-LANLayer2LLC BEHAVIOUR

MSOBV-LANLayer2LLC BEHAVIOUR
DEFINED AS

This object class defines the LLC for LANs. The operation of the LLC Entity is defined in the IEEE 802.2 Standard ;

21.7.18 .MSOBV-LayerWithCounters BEHAVIOUR

DEFINED AS

Individual classes derived from MSOOB-LayerWithCounters must specify within their behaviours the counter set defined for that class. There are two aspects to this specification:

- Which counters are included. This information is specified by identifying a list of values within Table 21-1 on page 21-43.
- Protocol-specific rules for incrementing each of the counters. The behaviour of the MSOOB-LSAPPAIRENTITY class, for example, would need to specify not only that the totalBytesTransmitted counter is included in its counter set, but also what the specific rules are for incrementing this counter: which header and trailer bytes are included in the count, are retransmitted bytes counted again, etc.

The counters themselves appear as instances of the MSOOB-Counter class, contained immediately below instances of the class in question. The class definition may identify certain counters as mandatory for that class, and others as optional ;

21.7.19 MSOBV-LibraryObject BEHAVIOUR

DEFINED AS

An instance of the MSOOB-LibraryObject class serves as a repository for reference objects, initial value objects, and other "utility" objects. The goal is to simplify the process of finding such objects. Ordinarily there will be exactly one MSOOB-LibraryObject instance for a managed system ;

21.7.20 MSOBV-LSAPEntity BEHAVIOUR

MSONT-LSAPEntity BEHAVIOUR
DEFINED AS
This object class manages the LLC LSAP Entity. The operation of LSAP Entity is defined in the 802.2 Standard ;

21.7.21 MSOBV-LSAPPairEntity BEHAVIOUR

MSOBV-LSAPPairEntity BEHAVIOUR
DEFINED AS
This object class manages the link connections within the LAN LLC. The operation of the LSAP Pair is defined in the IEEE 802.2 Standard ;

21.7.22 MSOBV-LSAPPairEntityAlarms BEHAVIOUR

MSOBV-LSAPPairEntityAlarms BEHAVIOUR
DEFINED AS

Attributes of interest in Alarms:
- All LSAP Pair attributes may be of interest in Alarms.

Alarm conditions:
- Loss of a logical link resulting from a remote link station sending segmented data.
- Loss of a logical link resulting from a remote link station not responding.
- Loss of a logical link resulting from a remote link station sending a U-format LPDU without the required information.
- Loss of a logical link resulting from a remote link station sending an invalid or unsupported command.
- Loss of a logical link resulting from a remote link station sending a frame with an invalid Transmitter Receive Sequence Number.
- Loss of a logical link resulting from a remote link station sending a frame with an I-field that was too short.
- Loss of a logical link resulting from a remote link station sending a frame with an I-field that was too long.
- Loss of a logical link resulting from a local link station sending a frame with an invalid Transmitter Receive Sequence Number.
- Loss of a logical link resulting from a local link station sending a frame with an I-field that was too long.
- Loss of a logical link resulting from a local link station sending an invalid or unsupported command.
- Loss of a logical link resulting from a local link station sending an invalid sequence number.
- Loss of a logical link resulting from a remote link station sending a Disconnect Mode response.
- Loss of a logical link resulting from a remote link station sending a Disconnect command.
- Loss of a logical link resulting from a remote link station sending a SABME command to a station which was already open.
- Loss of a logical link resulting from a remote link station sending an unexpected UA or RR.
- Loss of a logical link resulting from a remote link station sending an XID out of sequence.

- Loss of a logical link resulting from a local link station sending a S or U format frame containing unexpected data ;

21.7.23 MSOBV-LSAPPairRoute BEHAVIOUR

MSOBV-LSAPPairRoute BEHAVIOUR
DEFINED AS
This is a conditional part of the LSAP Pair Object. It is present if source routing is used and the LSAP Pair is cognizant of its route. This conditional package contains one attribute, the source route that the LSAP Pair uses to communicate. Source Routing is described in the IEEE 802.1D Standard ;

21.7.24 MSOBV-LSAPType3 BEHAVIOUR

MSOBV-LSAPType3 BEHAVIOUR
DEFINED AS
This is a conditional part of the LSAPEntity object. It exists if the LSAP object supports LLC Type 3 service. The operation of LLC Type 3 is defined by the IEEE 802.2 standard ;

21.7.25 MSOBV-MaxSampleInterval BEHAVIOUR

MSOBV-MaxSampleInterval BEHAVIOUR
DEFINED AS

The MSOAT-MaxSampleInterval allows a manager to specify a maximum interval for sampling at an agent. An agent is free to sample at the rate specified, or at any faster rate that it chooses. This rate refers to the period between consecutive determinations of counters' current values, for comparison with threshold comparison values ;

21.7.26 MSOBV-NameManagement BEHAVIOUR

MSOBV-NameManagement BEHAVIOUR
DEFINED AS
This object class provides the management of the Dynamic Address Resolution and Route Discovery protocols.
See Chapter 6, "Dynamic Address Resolution and Route Discovery" on page 6-1 ;

21.7.27 MSOBV-Reinitialize BEHAVIOUR

MSOBV-Reinitialize BEHAVIOUR
DEFINED AS
This CONFIRMED action causes all connections to be disconnected, all LSAPs to be deactivated, and the entire LLC sublayer to be reset to its initial configuration ;

21.7.28 MSOBV-RegisterRequestBehavior BEHAVIOUR

MSOBV-RegisterRequestBehavior BEHAVIOUR
DEFINED AS
This confirmed action is used by the LAN Manager to request that a LAN Station Manager register itself with the LAN Manager ;

21.7.29 MSOBV-RegisterCheck BEHAVIOUR

MSOBV-RegisterCheck BEHAVIOUR
DEFINED AS
This confirmed action is used by the LAN Manager to check the registration status of a group function/qualifier pair. The action may be sent to a functional address or an individual address. The response contains a list of managing processes with which the entity is registered. Only station managers which are managing that group function and qualifier pair respond to the Register Check. The Register Check frame contains a response type field which indicates when a receiving station manager should respond, for example, respond when registered, respond when not registered, respond regardless of qualifier value ;

:

21.7.30 MSOBV-ResourceManagement BEHAVIOUR

MSOBV-ResourceManagement BEHAVIOUR
DEFINED AS
This object class provides coordination of local management of LAN functions. It keeps the managing process information and routes the CMIP commands and responses. This object class is responsible for registration. It issues Function Present events and responds to the Register Requests, Register Checks and Deregister Requests ;

21.7.31 MSOBV-RPUEnable BEHAVIOUR

MSOBV-RPUEnable BEHAVIOUR
DEFINED AS
This UNCONFIRMED action is causes the receiver to begin remote program update process ;

:

21.7.32 MSOBV-SoftReset BEHAVIOUR

MSOBV-SoftReset BEHAVIOUR
DEFINED AS
This UNCONFIRMED action is to cause the receiver to reset its function. The implications of soft reset are documented in the object class behaviours that contain this action. ;

21.7.33 MSOBV-ThresholdControl BEHAVIOUR

MSOBV-ThresholdControl BEHAVIOUR
DEFINED AS

Creation Behaviour

Objects with this behaviour support creation via an initial value object, as follows:

- If either of the attributes MSOAT-MaxSampleInterval or MSOAT-TriggerList is not specified explicitly on the CREATE, creation via an initial value object is attempted.
- The initial value object, if it exists, is found under the system's MSOOB-LibraryObject instance. A search is made under the MSOOB-LibraryObject, for an object in which the value of the MSOAT-PolicyTargetScope attribute matches the ObjectClass of the object that will contain the object being created. If no match is found, creation is not done via an initial value object.
- If there is a match, values for the MSOAT-MaxSampleInterval and/or the MSOAT-TriggerList attributes are copied into the object being created.
- If creation was done via an initial value object, then the ObjectInstance identifier for the initial value object is copied into the MSOAT-ThresholdPolicyPointer attribute of the object being created.

Ongoing Behaviour

This behaviour is defined in terms of the following parameters:

Common Parameters:

SW	report switch—this is the value of the MSOAT-ReportSwitch attribute
maxSampInterval	maximum sample interval—this is the value of the MSOAT-MaxSampleInterval attribute
Interval Start Values	Set of (counterName, intervalStartValue), one for each counter in the counter set

Per-Trigger Parameters:

Numerator Parameters (always present for a trigger):

cName(n)*	name of the numerator counter
O(n)*	offset value for the numerator counter
C(n)	comparison value for the numerator counter
PW(n)	partial wrap indicator for the numerator counter
QW(n)	quick wrap indicator for the numerator counter

ERStartValue(n) value of the numerator counter at the last effective reset

* Visible externally in the MSOAT-TriggerList attribute

Denominator Parameters (present only if a trigger has a denominator counter):

cName(d)*	name of the denominator counter
O(d)*	offset value for the denominator counter
C(d)	comparison value for the denominator counter

PW(d) partial wrap indicator for the denominator counter

QW(d) quick wrap indicator for the denominator counter

ERStartValue(d) value of the denominator counter at the last effective reset

* Visible externally in the MSOAT-TriggerList attribute

Parameters Available from the MSOOB-Counter Objects:

c(x) current value of the counter

maxValue(x) the maximum value of the counter

```
** The following values are specified when the MSOOB-ThresholdControl
** object is created:
**      SW
**      maxSampleInterval
**
** The following values are sepcified when a trigger is added
** to the trigger list, or when a non-empty trigger list is
** specified at the creation of the MSOOB_ThrehsoldControl object:
**      cName(n)
**      O(n)
**      [cName(d)]
**      [O(d)]
** The denominator elements are not specified for a trigger defined
** to emit interval reports.
**
** Note:  It is not very clear in the typeface used here, but
** the following behaviour representation uses O(x), i.e., the offset
** value for the counter x, extensively.  Zero ('0') does not appear
** at all.


** At creation of the MSOOB-ThresholdControl object:

     For each counter in the counter set:
       intervalStartValue(x) := c(x)


** For each trigger in the initial trigger list when the
** MSOOBThresholdControl object is created, or when a new
** trigger is added:

     For each counter in the pair (i.e., for x = n [and x = d]):
        if c(x) + O(x) <= maxValue(x)
           then
             begin
               ** the counter will not be in partial wrap
               C(x) := c(x) + O(x)
               PW(x) := off
             end
           else
             begin
               ** the counter will be in partial wrap
               C(x) := c(x) - (maxValue(x) + 1) + O(x)
               PW(x) := on
             end
```

```
** When a trigger's numerator or denominator counter wraps:

   For x = n or x = d          ** depending on whether it was the numerator
                               ** or denominator counter that wrapped
     if PW(x) = off
       then
         ** If the threshold has not previously wrapped, then the quick
         ** wrap flag is set on
         QW(x) := on
       else
         ** If the threshold has already wrapped, then when the counter
         ** also wraps the partial wrap flag is set off
         PW(x) := off


** When any counter in the counter set wraps:

   begin
     emit MSONT-CounterSetReport notification
     for each counter in the counter set:
       intervalStartValue(x) := c(x)
   end


** When the current value of a trigger's numerator counter is checked.
** The frequency with which counters' values are checked is governed by
** the value of maxSampleInterval.

   If Comparison(n)
     then
       if SW=on
         then
           begin
             emit MSONT-CounterSetReport notification
             for each counter in the counter set:
               intervalStartValue(x) := c(x)
           end
       effectiveReset
     end


** When the current value of a trigger's denominator counter is checked.
** The frequency with which counters' values are checked is governed by
** the value of maxSampleInterval.

   If Comparison(d)
     then effectiveReset


** At deletion of the MSOOB-ThresholdControl object:

   emit MSONT-CounterSetReport notification
```

```
**************************************************************************
** Supporting boolean-valued function Comparison(x):

   Comparison(x) :=
      (PW(x)=off and c(x) >= C(x))  ** both count and comparison values are on
                                    ** the same "pass" through the counter,
                                    ** so it is valid to compare them
       or QW(x)=on                  ** if the count value is one "pass" ahead
                                    ** of the comparison value, the counter
                                    ** must have exceeded the threshold


** Supporting procedure effectiveReset:

   For each counter in the pair (i.e., for x = n [and x = d]):
     begin
       ** Increase the comparison value by adding the offset
       ** to the current count value.  If the comparison value
       ** wraps, set the comparison value to the appropriate
       ** distance from 0, and set the partial wrap flag.
       if c(x) + O(x) <= maxValue(x)
         then
           C(x) := c(x) + O(x)
         else
           begin
             C(x) := c(x) - (maxValue(x) + 1) + O(x)
             if QW(x)=off
               then
                 PW(x) := on
           end
       ** reset the quick wrap flag, since a counter cannot be in the
       ** quick wrap state after an effective reset
       QW(x) := off
       ** update the effective reset start value
       ERStartValue(x) := c(x)
     end ;
```

21.7.34 MSOBV-ThresholdControlInitialValues BEHAVIOUR

MSOBV-ThresholdControlInitialValues BEHAVIOUR
DEFINED AS

An object with this behaviour is a passive repository of initial values for the MSOAT-MaxSampleInterval and MSOAT-TriggerList attributes. ;

21.7.35 MSOBV-TRMACAlarms BEHAVIOUR

MSOBV-TRMACAlarms BEHAVIOUR
DEFINED AS

Attributes of interest in Alarms:
- All Token-Ring Layer 2 MAC Attributes may be of interest in Alarms.

Alarm conditions:
- Detection of a beaconing condition on the ring during the

insertion process.
- Adapter leaving the ring as part of the beacon automatic-recovery process. That is, the reporting station's adapter was a member of the beacon fault domain and removed itself from the ring to perform a self test, which was unsuccessful.
- Ring beaconing for a time longer than the hard-error detection timer. Manual intervention is required to recover the ring.
- Ring in a beaconing condition for a time shorter than the hard-error detection timer. When the stations in the beacon fault domain were queried, one or both of them had left the ring.
- Ring in a beaconing condition for less than 52 seconds and then recovered. The emitter of the Alarm either knows that neither station in the fault domain left the ring, or has no knowledge about whether a station removed itself from the ring in order to bypass the fault. ;

21.7.36 MSOBV-TokenRingLayer1 BEHAVIOUR

MSOBV-TokenRingLayer1 BEHAVIOUR
DEFINED AS
This object class defines the physical layer information pertaining to a Token-Ring adapter. Operation of the Token-Ring layer 1 is defined in the ISO 8802-5 Token-ring Standard.

21.7.37 MSOBV-TokenRingLayer2MAC BEHAVIOUR

MSOBV-TokenRingLayer2MAC BEHAVIOUR
DEFINED AS
This object class defines the MAC sublayer for token ring. The operation of the Token-Ring MAC is defined in the 802.5 Standard. ;

21.8 Attribute Values Tables

This chapter contains a number of tables, each associated with a specific IBM-registered attribute. The syntax for each of these attributes specifies a fixed-size octet string, containing a code point. The tables in this chapter associate each of the code points for an attribute with several pieces of information:

- **Value Name:** The name of the attribute value associated with the code point. This is a part of the specification of the value; it is thus not available for national language translation.
- **Text Description:** A text description, suitable for display, corresponding to this name. This text is available for national language translation for display at a manager.
- **Definition:** A definition of the attribute value. This text is available for national language translation for display at a manager.
- Possibly other pieces of information unique to a specific attribute.

21.8.1 Values of the MSOAT-CounterName ATTRIBUTE

In addition to the information described above which is common to all attribute value tables defined in this chapter, this table contains two elements unique to the CounterName attribute:

- **Summary Role:** An indication of the role the counter plays in counter summaries at a Problem (Fault?) Management focal point. The following summary roles are defined:

  Transmit errors The counter is added into two summary counts: total transmit errors and total errors.

  Receive errors The counter is added into two summary counts: total receive errors and total errors.

  Total errors Since the counter cannot be associated with either the transmit or the receive direction, it is added into only one summary count: total errors.

  Transmit traffic The counter is added into one summary count: total transmit traffic.

  Receive traffic The counter is added into one summary count: total receive traffic.

  None The counter is not added into any summary count.

- **Counter Sets:** A list of the architecturally-defined counter sets to which the counter belongs. The specifications for these architectures provide the detailed rules for determining when the counter should be incremented in each environment.

Table 21-1 (Page 1 of 6). Values for the CounterName Attribute

Code Point			Counter Sets
X'0011'	**Value Name**	abortedFramesTransmitted	TR LAN MAC
	Text Description	Aborted Frames Transmitted	
	Definition	The number of incomplete Data Link Control (DLC) frames transmitted	
	Summary Role	Transmit errors	
X'0012'	**Value Name**	misaddressedFramesReceived	LAN LLC
	Text Description	Misaddressed Frames Received	
	Definition	The number of frames received correctly, but for which no active station exists, therefore these frames cannot be routed	
	Summary Role	Receive errors	
X'0015'	**Value Name**	totalFramesTransmitted	LAN LLC LAN LSAP pair
	Text Description	Total Frames Transmitted	
	Definition	The sum of the number of Information frames, Unnumbered Information frames, and Supervisory frames transmitted	
	Summary Role	Transmit traffic	

Table 21-1 (Page 2 of 6). Values for the CounterName Attribute

Code Point			Counter Sets
X'0016'	**Value Name**	totalFramesReceived	LAN LLC LAN LSAP pair
	Text Description	Total Frames Received	
	Definition	The sum of the number of Information frames, Unnumbered Information frames, and Supervisory frames received	
	Summary Role	Receive traffic	
X'0017'	**Value Name**	pdusRetransmitted receiveSEQErrors	LAN LLC LAN LSAP pair
	Text Description	PDUs Retransmitted	
	Definition	The number of protocol data units retransmitted as a result of time out on the link or any other protocol errors, such as sequence errors	
	Summary Role	Transmit errors	
X'0019'	**Value Name**	totalBytesTransmitted	LAN LLC LAN LSAP pair
	Text Description	Total Bytes Transmitted	
	Definition	The total number of bytes transmitted by a layer entity	
	Summary Role	*none*	
X'001A'	**Value Name**	totalBytesReceived	LAN LLC LAN LSAP pair
	Text Description	Total Bytes Received	
	Definition	The total number of bytes received by a layer entity	
	Summary Role	*none*	
X'001B'	**Value Name**	totalBytesRetransmitted	LAN LLC LAN LSAP pair
	Text Description	Total Bytes Retransmitted	
	Definition	The total number of bytes retransmitted by a layer entity due to transmission errors	
	Summary Role	*none*	
X'0020'	**Value Name**	iFramesTransmitted	LAN LLC LAN LSAP pair
	Text Description	Information Frames Transmitted	
	Definition	The number of (numbered) information frames transmitted	
	Summary Role	Transmit traffic	

Table 21-1 (Page 3 of 6). Values for the CounterName Attribute

Code Point			Counter Sets
X'0021'	**Value Name**	iFramesReceived	LAN LLC LAN LSAP pair
	Text Description	Information Frames Received	
	Definition	The number of (numbered) information frames received	
	Summary Role	Receive traffic	
X'0022'	**Value Name**	PDUsDiscarded	LAN LLC LAN LSAP
	Text Description	PDUs Discarded resources.	
	Definition	The number of received PDUs discarded due to insufficient resources	
	Summary Role	Receive errors	
X'0023'	**Value Name**	totalConnections	LAN LSAP
	Text Description	Total Connections	
	Definition	The number of successful connections established by this LSAP since the it was last activated	
	Summary Role	*none*	
X'0024'	**Value Name**	uiFramesTransmitted	LAN LSAP
	Text Description	Unnumbered Information Frames Transmitted	
	Definition	The number of unnumbered information frames transmitted	
	Summary Role	Transmit traffic	
X'0025'	**Value Name**	uiFramesReceived	LAN LSAP
	Text Description	Unnumbered Information Frames Received	
	Definition	The number of unnumbered information frames received	
	Summary Role	Receive traffic	
X'0026'	**Value Name**	lanType3FramesTransmitted	LAN LSAP
	Text Description	LAN Type 3 Frames Transmitted	
	Definition	The number of acknowledged connectionless information PDUs transmitted	
	Summary Role	Transmit traffic	
X'0027'	**Value Name**	type3FramesReceived	LAN LSAP
	Text Description	LAN Type 3 Frames Received	
	Definition	The number of acknowledged connectionless information PDUs received	
	Summary Role	Receive traffic	

Table 21-1 (Page 4 of 6). Values for the CounterName Attribute

Code Point			Counter Sets
X'0028'	**Value Name**	lanType3FramesRetransmitted	LAN LSAP
	Text Description	LAN Type 3 Frames Retransmitted	
	Definition	The number of acknowledged connectionless information PDUs retransmitted (at least once) by the Type 3 LLC service	
	Summary Role	Transmit errors	
X'0029'	**Value Name**	lanType2AcknowledgmentTimerTimeouts	LAN LSAP pair
	Text Description	LAN Type 2 Acknowledgment Timer Timeouts	
	Definition	The number of times the Ack timer has expired	
	Summary Role	*none*	
X'002A'	**Value Name**	localBusyOccurrences	LAN LSAP pair
	Text Description	Local Busy Occurrences	
	Definition	The number of busy occurrences for the local LSAP connection entity	
	Summary Role	*none*	
X'002B'	**Value Name**	tokenRingMACLineErrors	TR LAN MAC
	Text Description	Token Ring MAC Line Errors	
	Definition	The number of code violations between the starting and ending delimiters and frame check sequence errors detected by this station	
	Summary Role	Receive errors	
X'002C'	**Value Name**	tokenRingMACBurstErrors	TR LAN MAC
	Text Description	Token Ring MAC Burst Errors	
	Definition	The number of times a station detects the absence of transitions for five half-bit times	
	Summary Role	Receive errors	
X'002D'	**Value Name**	tokenRingMACACErrors	TR LAN MAC
	Text Description	Token Ring MAC A/C Errors	
	Definition	The number of times a station receives an AMP or SMP frame in which A and C are equal to 0, and then receives another SMP frame with A and C equal to 0 without first receiving an AMP frame.	
	Summary Role	Receive errors	

Table 21-1 (Page 5 of 6). Values for the CounterName Attribute

Code Point			Counter Sets
X'002E'	**Value Name**	tokenRingMACInternalErrors	TR LAN MAC
	Text Description	Token Ring MAC Internal Errors	
	Definition	The number of recoverable internal errors detected by a token ring MAC station	
	Summary Role	*none*	
X'002F'	**Value Name**	tokenRingMACLostFrameErrors	TR LAN MAC
	Text Description	Token Ring MAC Lost Frame Errors	
	Definition	The number of times frames transmitted by a station fail to return to it (and thus the active monitor must issue a new token)	
	Summary Role	Transmit errors	
X'0030'	**Value Name**	tokenRingMACReceiveCongestionErrors	TR LAN MAC
	Text Description	Token Ring MAC Receive Congestion Errors	
	Definition	The number of times a station recognizes a frame addressed to its specific address, but has no available buffer space for the frame.	
	Summary Role	Receive errors	
X'0031'	**Value Name**	tokenRingMACFrameCopiedErrors	TR LAN MAC
	Text Description	Token Ring MAC Frame-Copied Errors	
	Definition	The number of times a station recognizes a frame addressed to its specific address and detects that the address recognized bits are set, indicating a possible duplicate address	
	Summary Role	Receive errors	
X'0032'	**Value Name**	tokenRingMACTokenErrors	TR LAN MAC
	Text Description	Token Ring MAC Token Errors	
	Definition	The number of times the active monitor detects an error condition resulting in the need to transmit a token	
	Summary Role	Receive errors	
X'0033'	**Value Name**	tokenRingMACFrequencyErrors	TR LAN MAC
	Text Description	Token Ring MAC Frequency Errors	
	Definition	The number of times a ring station detects a frequency error	
	Summary Role	Receive errors	

Table 21-1 (Page 6 of 6). Values for the CounterName Attribute

Code Point			Counter Sets
X'0034'	**Value Name**	unrecognizedPDUs	LAN LLC
	Text Description	Unrecognized PDUs	
	Definition	The number of received PDUs having an unrecognized value in the control field	
	Summary Role	Receive errors	
X'0035'	**Value Name**	testCommandsReceived	LAN LSAP
	Text Description	Test Commands Received	
	Definition	The number of TEST commands received	
	Summary Role	Receive traffic	
X'0036'	**Value Name**	testResponsesTransmitted	LAN LSAP
	Text Description	Test Responses Transmitted	
	Definition	The number of TEST responses transmitted	
	Summary Role	Transmit traffic	
X'0037'	**Value Name**	timer	*All*
	Text Description	Timer (Milliseconds)	
	Definition	Timer, in milliseconds; in order to preclude excessive wrap reports, this counter must be at least 4 octets in size in all counter sets	
	Summary Role	*none*	

Chapter 22. Alarm and Alert Definitions

This document defines the new alerts and alarms that will be emitted either by the LAN Station Manager or by the LAN Manager.

The LAN LLC Alerts 1 - 11, and Token-Ring LAN alerts are existing alerts that are documented in the *Systems Network Architecture Management Services Reference*.

22.1 Alarms emitted by the Token-Ring Layer 2 MAC objects

The following alarms are sent from the LAN Station Manager to a manager that is not connected via the token-ring (beaconing alarms should not be sent over a token-ring).

Token-Ring Layer 2 MAC Alarm 1

Alarm Condition: The adapter detected a beaconing condition on the ring during the insertion process. The insertion process did not complete.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	Token-Ring Layer 2 MAC object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'3703' (1)	CommunicationAlarm [3] Open Failure Token-ring fault domain Critical
Monitored Attributes	(Object ID)	SegmentNumber
Proposed Repair Action	X'3101'	Contact the token-ring administrator responsible for this LAN
Problem Data Probable Causes	X'3703'	Token-ring fault domain
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'3703'	Token-ring fault domain
Actions	X'1009' X'3301' X'2010' X'3101'	Attempt to re-open the adapter after 30 seconds If problem persists then do the following Review link detailed data Contact token-ring administrator responsible for this LAN
Additional Data	(51) SV (06) SF (26) SF (07) SF	LAN LCS Data Token-ring fault domain description Fault domain names (optional) Beacon data

Notes:

1. The corresponding Alert is Token-Ring LAN Alert 2
2. Object Instance identifying this MAC
3. Defined in OSI Standard 10165-2

Token-Ring Layer 2 MAC Alarm 2

Alarm Condition: The reporting station's adapter has left the ring as part of the beacon automatic-recovery process. That is, the reporting station's adapter was a member of the beacon fault domain and removed itself from the ring to perform a self test, which was unsuccessful.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	Token-Ring Layer 2 MAC object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'3702' (1)	CommunicationAlarm [3] Auto-removal Token-ring lobe Critical
Monitored Attributes	(Object ID)	SegmentNumber
Proposed Repair Action	X'3101'	Contact the token-ring administrator responsible for this LAN.
Problem Data Probable Causes	 X'3702'	 Token-ring lobe
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'3320' X'3711' X'3434'	Local token-ring adapter Local access unit Local lobe cables
Actions	X'2010' X'3101' X'0105'	Review link detailed data Contact token-ring administrator responsible for this LAN Request verification of management server reporting links[4]
Additional Data	(51) SV (23) SF	LAN LCS Data (optional) Local Individual MAC Name (optional)

Notes:

1. The corresponding Alert is Token-Ring LAN Alert 7
2. Object Instance identifying this MAC
3. Defined in OSI Standard 10165-2
4. This code point is present if the sending product is a LAN manager and has reporting links with remote management servers.

Token-Ring Layer 2 MAC Alarm 3

Alarm Condition: The ring has been beaconing for a time longer than the hard-error detection timer. Manual intervention is necessary to recover the ring.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	Token-Ring Layer 2 MAC object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'3703' (1)	CommunicationAlarm [3] Token-ring inoperative Token-ring fault domain Critical
Monitored Attributes	(Object ID)	SegmentNumber
Proposed Repair Action	X'3101'	Contact the token-ring administrator responsible for this LAN.
Problem Data Probable Causes	 X'3703'	 Token-ring fault domain
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'3703'	Token-ring fault domain
Actions	X'2010' X'3101' X'0105'	Review link detailed data Contact token-ring administrator responsible for this LAN Request verification of management server reporting links[4]
Additional Data	(51) SV (06) SF (26) SF (07) SF	LAN LCS Data Token-ring fault domain description Fault domain names (optional) Beacon data

Notes:

1. The corresponding Alert is Token-Ring LAN Alert 9
2. Object Instance identifying this MAC
3. Defined in OSI Standard 10165-2
4. This code point is present if the sending product is a LAN manager and has reporting links with remote management servers.

Token-Ring Layer 2 MAC Alarm 4

Alarm Condition: The ring was in a beaconing condition for a time shorter that the hard-error detection timer. When the stations in the beacon fault domain were queried, one or both of them had left the ring.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	Token-Ring Layer 2 MAC object class
Object Instance	Distinguished Name [2]	

Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'3714' (1)	CommunicationAlarm [3] Auto removal Remote token-ring lobe Critical
Monitored Attributes	(Object ID)	SegmentNumber
Proposed Repair Action	X'3101'	Contact the token-ring administrator responsible for this LAN.
Problem Data Probable Causes	X'3714'	Remote token-ring lobe
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'3321' X'3713' X'3435'	Remote token-ring adapter Remote access unit Remote lobe cables
Actions –	X'2010' X'3101' X'0105'	Review link detailed data Contact token-ring administrator responsible for this LAN Request verification of management server reporting links[4]
Additional Data	(51) SV (06) SF (26) SF (08) SF (28) SF	LAN LCS Data Token-ring fault domain description[5] Fault domain names (optional)[6] Single Individual MAC Address[7] Single Individual MAC Name[8]

Notes:

1. The corresponding Alert is Token-Ring LAN Alert 10
2. Object Instance identifying this MAC
3. Defined in OSI Standard 10165-2
4. This code point is present if the sending product is a LAN manager and has reporting links with remote management servers.
5. This subfield is present if the sending product has determined that both beacon fault domain stations left the ring as part of the automatic recovery process.
6. This subfield is optionally present, but only present if the sending product has determined that both beacon fault domain stations left the ring as part of the automatic recovery process.
7. This subfield is present if the sending product has determined that only one of the beacon fault domain stations left the ring as part of the automatic recovery process.
8. This subfield is optionally present, but only if the sending product has determined only one of the beacon fault domain stations left the ring as part of the automatic recovery process.

Token-Ring Layer 2 MAC Alarm 5

Alarm Condition: The ring was in a beaconing condition for less than 52 seconds and then recovered. The sender of this Alarm either knows that neither station in the fault domain left the ring, or has no knowledge about whether a station removed itself from the ring in order to bypass the fault.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	Token-Ring Layer 2 MAC object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'3703' (1)	CommunicationAlarm [3] Token-ring temporary error Token-ring fault domain Critical
Monitored Attributes	(Object ID)	SegmentNumber
Proposed Repair Action	X'3101'	Contact the token-ring administrator responsible for this LAN.
Problem Data Probable Causes	 X'3703'	 Token-ring fault domain
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'3703'	Token-ring fault domain
Actions	X'2010' X'3101' X'0105'	Review link detailed data Contact token-ring administrator responsible for this LAN Request verification of management server reporting links[4]
Additional Data	(51) SV (06) SF (26) SF (07) SF	LAN LCS Data Token-ring fault domain description Fault domain names (optional) Beacon data

Notes:

1. The corresponding Alert is Token-Ring LAN Alert 11
2. Object Instance identifying this MAC
3. Defined in OSI Standard 10165-2
4. This code point is present if the sending product is a LAN manager and has reporting links with remote management servers.

22.2 Alarms emitted by the LSAP Pair Entity objects

The following alarms are sent from the LAN Station Manager to the LAN Manager.

LSAP Pair Entity Alarm 1

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station has sent an unexpected frame containing segmented data. This resulted in a Frame Reject response. [1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'1023' (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in remote node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'1023' X'2007'	 Communications program in remote node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'1023' X'F0xx' [5]	Communications program in remote node Segmented data not expected
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 12
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge
5. New code point to be assigned

LSAP Pair Entity Alarm 2

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station does not respond. The inactivity timer (Ti) or acknowledgment timer (T1) has expired, causing the remote station to be polled. The remote station does not respond to the poll. [1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'2107' (1)	CommunicationAlarm [3] Link error LAN LLC communications/remote node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'2107'	 LAN LLC communications/remote node
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'2107' X'F017'	LAN LLC communications/remote node Poll count exhausted
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for this LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 1
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge

LSAP Pair Entity Alarm 3

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station has sent a U-format LPDU without the required information. This resulted in a Frame Reject response. [1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'1023' (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in remote node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tITimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'1023' X'2007'	 Communications program in remote node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'1023' X'F0xx' [5]	Communications program in remote node U-format LPDU missing data was received
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 13
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge
5. New code point to be assigned

LSAP Pair Entity Alarm 4

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station sent an invalid or unsupported command or response to the local link station. This resulted in the local link station returning a Frame Reject response.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'1023' (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in remote node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'1023' X'2007'	 Communications program in remote node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'1023' X'F020'	Communications program in remote node Invalid/unsupported command or response received
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 8
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge

LSAP Pair Entity Alarm 5

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station sent a frame with an invalid N(r). This resulted in a Frame Reject response. [1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'1023' (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in remote node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tITimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'1023' X'2007'	 Communications program in remote node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'1023' X'F022'	Communications program in remote node Invalid N(r) received
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 10
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge

LSAP Pair Entity Alarm 6

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station sent a frame with an I-field that was too short. This resulted in a Frame Reject response. [1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'1023' (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in remote node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'1023' X'2007'	 Communications program in remote node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'1023' X'F0xx' [5]	Communications program in remote node Received I-field too short
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 14
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge
5. New code point to be assigned

LSAP Pair Entity Alarm 7

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station sent a frame with an I-field that was too long. This resulted in a Frame Reject response. [1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'1023' (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in remote node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'1023' X'2007'	 Communications program in remote node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'1023' X'F023'	Communications program in remote node Received I-field exceeded maximum length
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 11
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge

LSAP Pair Entity Alarm 8

Alarm/Alert Condition: A LAN logical link has been lost. The local link station sent a frame with an invalid N(r). This resulted in the remote link station returning a Frame Reject response.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'10xx' (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in local node[4] Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [5]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'10xx' [4] X'2007'	 Communications program in local node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'10xx' [4] X'F012'	Communications program in local node Frame reject received: invalid N(r) sent
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 6
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. New code point to be assigned
5. This attribute is present if the lost logical link traversed a MAC bridge

LSAP Pair Entity Alarm 9

Alarm/Alert Condition: A LAN logical link has been lost. The local link station sent a frame with an I-field that was too long. This resulted in the remote link station returning a Frame Reject response.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'10xx' (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in local node[5] Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tITimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'10xx' [5] X'2007'	 Communications program in local node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'10xx' [5] X'F013'	Communications program in local node Frame reject received: maximum I-field length exceeded
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN

Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 7
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge
5. New code point to be assigned

LSAP Pair Entity Alarm 10

Alarm/Alert Condition: A LAN logical link has been lost. The local link station sent an invalid or unsupported command or response to the remote link station. This resulted in the remote link station returning a Frame Reject response.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'10xx' [5] (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in local node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'10xx' [5] X'2007'	 Communications program in local node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'10xx' [5] X'F010'	Communications program in local node Frame reject received: Invalid/unsupported command or response sent

Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 4
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge
5. New code point to be assigned

LSAP Pair Entity Alarm 11

Alarm/Alert Condition: A LAN logical link has been lost. The local link station sent an invalid sequence number to the remote link station. This resulted in the remote link station returning a Reject.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'10xx' [5] (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in local node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'10xx' [5] X'2007'	 Communications program in remote node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	

Failure Causes	X'10xx' [5] X'F0xx' [5]	Communications program in remote node Reject received: invalid N(s) sent
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 15
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge
5. New code point to be assigned

LSAP Pair Entity Alarm 12

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station sent a Disconnect Mode response to the local link station.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'2007' (1)	CommunicationAlarm [3] Link error LAN LLC communications Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	X'2007'	LAN LLC communications
User Causes	(none)	
Install Causes	(none)	

Failure Causes	X'2007' X'F01A'	LAN LLC communications DM received
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 2
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge

LSAP Pair Entity Alarm 13

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station sent a Disconnect command to the local link station.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'2007' (1)	CommunicationAlarm [3] Link error LAN LLC communications Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tITimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	X'2007'	LAN LLC communications
User Causes	(none)	
Install Causes	(none)	

Failure Causes	X'2007' X'F0xx' [5]	LAN LLC communications DISC received
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 16
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge
5. New code point to be assigned

LSAP Pair Entity Alarm 14

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station sent a SABME command to the local link station which was already open (had been initialized via a SABME-UA exchange).[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'1023' (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in remote node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'1023' X'2007'	 Communications program in remote node LAN LLC Communications

User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'1023' X'F016'	Communications program in remote node SABME received while in ABME
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 3
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge

LSAP Pair Entity Alarm 15

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station sent an unexpected UA or RR to the local link station. This resulted in a Frame Reject response. [1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'1023' (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in remote node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN

Problem Data Probable Causes	 X'1023' X'2007'	 Communications program in remote node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'1023' X'F0xx' [5]	Communications program in remote node Unexpected UA or RR received
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 17
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge
5. New code point to be assigned

LSAP Pair Entity Alarm 16

Alarm/Alert Condition: A LAN logical link has been lost. The remote link station sent a XID out of sequence to the local link station.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'1023' (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in remote node Critical
Monitored Attributes	(Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID) (Object ID)	LSAPPairId K RW MaximumRetransmissions t2TimerValue t1TimerValue tlTimerValue MaxOutIncrement AccessPriority route (Optional) [4]

Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'1023' X'2007'	 Communications program in remote node LAN LLC communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'2007' X'F0xx' [5]	LAN LLC communications Received XID out of sequence
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 18
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge
5. New code point to be assigned

LSAP Pair Entity Alarm 17

Alarm/Alert Condition: A LAN logical link has been lost. The local link station sent a S or U format frame containing unexpected data to the remote link station. This resulted in the remote link station returning a Frame Reject response.[1]

Mode	(0)	Event Report (unconfirmed)
Object Class	(Object ID)	LSAP Pair object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'10xx' [5] (1)	CommunicationAlarm [3] LAN LLC protocol error Communications program in local node Critical

Monitored Attributes	(Object ID)	LSAPPairId
	(Object ID)	K
	(Object ID)	RW
	(Object ID)	MaximumRetransmissions
	(Object ID)	t2TimerValue
	(Object ID)	t1TimerValue
	(Object ID)	tlTimerValue
	(Object ID)	MaxOutIncrement
	(Object ID)	AccessPriority
	(Object ID)	route (Optional) [4]
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	X'10xx' [5] X'2007'	Communications program in local node LAN LLC Communications
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'10xx' [5] X'F0xx' [5]	Communications program in local node Frame reject received: S or U format frame containing unexpected data received
Actions	X'3300' X'2010' X'3103'	If problem reoccurs then do the following: Review link detail data Contact LAN administrator responsible for the LAN
Additional Data	(51) SV (02) SF (23) SF (24) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Name (Optional) Remote Individual MAC Name (Optional)

Notes:

1. The corresponding Alert is LAN LLC Alert 19
2. Object Instance identifying this LSAP Pair
3. Defined in OSI Standard 10165-2
4. This attribute is present if the lost logical link traversed a MAC bridge
5. New code point to be assigned

22.3 Alarms emitted by the CAU objects

The following alarms are sent from the LAN Station Manager CAU to the LAN Manager (as opposed to sending Events for these conditions).

If an Alarm is emitted for a problem resolution, the severity that is issued (the Perceived Severity field) should have the value (5) to indicated that a problem has cleared. The Correlated Notifications field should also be used to give the Notification ID of the Alarm that was emitted for the original problem.

If it is not possible for the Alarm emitter to know whether the Alarm is for a problem or a problem resolution, the indeterminate severity should be used.

CAU Alarm 1

Alarm/Alert Condition: The status has changed for the CAU back-up path. This Alarm will be emitted to report a problem (perceived severity 4) and a problem resolution (perceived severity 5).[1]

Mode	(1)	Event Report (Confirmed)
Object Class	(Object ID)	CAU object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'3701' (4)	CommunicationAlarm [3] Back-up path status change Token-Ring LAN component Warning
Monitored Attributes	(Object ID)	CAUVitalInfo
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'3701' X'3703'	 Token-Ring LAN component Token-Ring fault domain
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'3701' X'F05C'	Token-Ring LAN component Token-Ring began or terminated beaconing
Recommended Actions	X'32A0' X'82' SF	Report the following (Back-up path status) - code point X'42'[4]
Additional Data	(51) SV (02) SF (06) SF	LAN LCS Data Ring/Segment Identifier Ring Fault Domain Description (Optional)[5]

Notes:

1. The corresponding Alert is LAN Access Unit Alert 1
2. Object Instance identifying this CAU
3. Defined in OSI Standard 10165-2
4. ASN.1 label = BPStatus
5. Ring fault domain description should be sent on problem reports, but is optional for problem resolution Alarms.

CAU Alarm 2

Alarm/Alert Condition: The wrap status for a CAU has changed.[1]

Mode	(1)	Event Report (Confirmed)
Object Class	(Object ID)	CAU object class

Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'3701' (0)	CommunicationAlarm [3] Path wrap status change Token-Ring LAN component Indeterminate
Monitored Attributes	(Object ID)	CAUVitalInfo
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'3701' X'3707'	 Token-Ring LAN component Token-Ring LAN cables
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'3711' X'3713' X'3707'	Local Access Unit Remote access unit Token-Ring LAN cables
Recommended Actions	X'0105' X'32A0' X'82' SF	Request verification of management server reporting links Report the following (Wrap status) - code point X'43' [4]
Additional Data	(51) SV (02) SF	LAN LCS Data Ring/Segment Identifier

Notes:

1. The corresponding Alert is LAN Access Unit Alert 2
2. Object Instance identifying this CAU
3. Defined in OSI Standard 10165-2
4. ASN.1 label = WrapType

CAU Alarm 3

Alarm/Alert Condition: The CAU base unit has detected an internal error. This Alarm will be emitted to report a problem (perceived severity 1) and a problem resolution (perceived severity 5).[1]

Mode	(1)	Event Report (Confirmed)
Object Class	(Object ID)	CAU object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'3751' (1)	CommunicationAlarm [3] LAN error Token-Ring access unit Critical

Monitored Attributes	(Object ID)	CAUVitalInfo
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	X'3751'	Token-Ring access unit
User Causes	(none)	
Install Causes	(none)	
Failure Causes	X'3752'	Base unit internal error
Recommended Actions	X'32A0' X'82' SF	Report the following (Error code) - code point X'07'[4]
Additional Data	(51) SV (02) SF	LAN LCS Data Ring/Segment Identifier

Notes:

1. The corresponding Alert is LAN Access Unit Alert 3
2. Object Instance identifying this CAU
3. Defined in OSI Standard 10165-2
4. ASN.1 label = DiagErrCode

CAU Alarm 4

Alarm/Alert Condition: There is a mismatch between the number of lobes active and addresses known in a CAU.[1]

Mode	(1)	Event Report (Confirmed)
Object Class	(Object ID)	CAU object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'3321' (0)	EnvironmentalAlarm [3] Unauthorized LAN insertion attempted Remote Token-Ring adapter Indeterminate
Monitored Attributes	(Object ID)	CAUVitalInfo
Proposed Repair Action	X'3002'	Contact the security control representative.
Problem Data Probable Causes	X'3321'	Remote Token-Ring adapter
User Causes	X'7105' X'7130' X'710C'	Unauthorized user attempted insertion into LAN Multiple adapters attached to one lobe Unauthorized trace tool in LAN

Recommended Actions	X'32A0' X'82' SF	Report the following (Configuration data) - code point X'45'[4]
Install Causes	(none)	
Failure Causes	(none)	
Additional Data	(51) SV (02) SF	LAN LCS Data Ring/Segment Identifier

Notes:

1. The corresponding Alert is LAN Access Unit Alert 4
2. Object Instance identifying this CAU
3. Defined in OSI Standard 10165-2
4. ASN.1 label = ConfigErrPresent

CAU Alarm 5

Alarm/Alert Condition: A CAU ignored a Force Remove command.[1]

Mode	(1)	Event Report (Confirmed)
Object Class	(Object ID)	CAU object class
Object Instance	Distinguished Name [2]	
Alarm Type Event Argument Probable Cause Specific Problem Perceived Severity	(Object ID) (Object ID) X'7003' (0)	CommunicationAlarm [3] Force Remove ignored Network operator Indeterminate
Monitored Attributes	(Object ID)	CAUVitalInfo
Proposed Repair Action	X'3103'	Contact the LAN administrator responsible for this LAN
Problem Data Probable Causes	 X'7003'	 Network operator
User Causes	X'7101'	Token-Ring remove adapter command received
Recommended Actions	X'3103'	Contact LAN administrator responsible for this LAN
Install Causes	(none)	
Failure Causes	(none)	
Additional Data	(51) SV (02) SF (03) SF (04) SF	LAN LCS Data Ring/Segment Identifier Local Individual MAC Address Remote Individual MAC Address (Optional)

Chapter 23. ASN.1 Definitions

23.1 LAN-COMMON

```
LAN-COMMON
 DEFINITIONS ::= BEGIN


 AccessUnitId ::= OCTET STRING (4)    -- used as distinguishing
                                      -- attribute for Function Present
                                      -- and attribute in Token-Ring
                                      -- layer 1 object class

 AccessUnitName ::= OCTET STRING (5)  -- Naming attribute in CAU

 AccessUnitNumber ::= CHOICE {NULL, GraphicString}

 AcknowledgementTime ::= INTEGER

 ActivatedTypes ::= LLCServices

 ActiveConnections ::= SET OF LLCRemoteAddress

 ActiveLSAPs ::= SET OF LSAP

 AccessPriority ::= OCTET STRING

 AdapterInterruptLevel ::= OCTET STRING (1)

 AdapterNumber ::= OCTET STRING

 AddrInsertType  ::= SEQUENCE OF MACAddress

 AMName ::= OCTET STRING (5)


 AMStatus ::= SEQUENCE {
       amnumber [0] IMPLICIT OCTET STRING (1),
       status [1] IMPLICIT StatusType }

 AMStatusEventData ::= SET OF SEQUENCE {
       amnumber [0] IMPLICIT OCTET STRING (1),
       status [1] IMPLICIT StatusType }

 AMStatusType ::= SEQUENCE {
    status     [0] IMPLICIT StatusType,
    lobesDeac  [1] IMPLICIT OCTET STRING(8),
    lobesInsert [2] IMPLICIT OCTET STRING(8),
    amNumber   [3] IMPLICIT OCTET STRING(1),
    numLobes   [4] IMPLICIT OCTET STRING(1) }

 ArchReleaseLevel ::= INTEGER
```

```
LLCStatus ::= SEQUENCE {
              [0] IMPLICIT LSAPPairId,
              [1] IMPLICIT K,
              [2] IMPLICIT RW,
              [3] IMPLICIT MaximumRetransmissions,
   t2TimerValue   [4] IMPLICIT Timer,
   t1TimerValue   [5] IMPLICIT Timer,
   tiTimerValue   [6] IMPLICIT Timer,
   maxOutIncrement [7] IMPLICIT Timer,
              [8] IMPLICIT AccessPriority,
   route          [9] IMPLICIT Route OPTIONAL }

LLCServices ::= BIT STRING {
            type1 (0),
            type2 (1),
            type3Initiate (2),
            type3ReceiveDat (3),
            type3ReturnData (4)  }


LobeId ::= OCTET STRING(1)

Location ::= CHOICE {NULL, GraphicString }

LobeId ::= OCTET STRING(1)

LobeStatus ::= SEQUENCE {
   amNumber    [0] IMPLICIT OCTET STRING(1),
   adapterLobe [1] IMPLICIT OCTET STRING(1),
   enableStatus [2] IMPLICIT LAN-COMMON.EnableType }

LobeStatusEventData ::= SET OF SEQUENCE {
   amNumber    [0] IMPLICIT OCTET STRING(1),
   adapterLobe [1] IMPLICIT OCTET STRING(1),
   enableStatus [2] IMPLICIT LAN-COMMON.EnableType,
   insertStatus [3] IMPLICIT LAN-COMMON.InsertType,
   addr        [4] IMPLICIT LAN-COMMON.MacAddress }

-- Manufacturer (logo) Identifier  Code

LSAP ::= OCTET STRING

LSAPPairId ::= SEQUENCE {
   localLSAP       [0] IMPLICIT LSAP,
   localAddress    [1] IMPLICIT MACAddress,
   remoteLSAP      [2] IMPLICIT LSAP,
   remoteAddress   [3] IMPLICIT MACAddress}

LSAPPairName ::= SEQUENCE {
          [0] IMPLICIT LSAP,     -- remote LSAP
          [1] IMPLICIT MACAddress -- remote MAC Address
                                  }
```

```
MACAddress ::= OCTET STRING (6)

MACDriverVersionNumber ::=OCTET STRING(2)
-- First octet represent major version (2 BCD digits)
-- Second octet represent minor version (2 BCD digits)

MACType ::= OCTET STRING (16)


MachineType ::= GraphicString
                    -- defined by products
ManagingProcessInfo ::= SEQUENCE {
 managingProcessTitle          [0] IMPLICIT DARRD.RegularIdentifier,
 managingProcessAddress        [1] IMPLICIT MACAddress,
 lostMPRetries                 [2] IMPLICIT Integer,
 associatedThresholds          [3] IMPLICIT SET OF LAN-COUNTER.ThresholdName }

ManufacturerID ::= OCTETSTRING -- 3 octets maximum
ManufacturerProductID ::= ANY
ManufacturerProductVersion ::= ANY

MaximumLinkStationsConfigured ::= INTEGER

MaximumPDUN3 ::= INTEGER

MaximumLLCInformationFieldSize ::= INTEGER

MaximumRetransmissions ::= INTEGER

MaximumLSAPSConfigured ::= INTEGER

MaxOutIncrement ::= INTEGER

MediaType ::= INTEGER {
          fiber (0),
          copper (1) }

MicrocodeLevel ::= OCTET STRING(1..32)

MulticastAddress ::= SET OF MACAddress
                                }
NMName ::= OCTET STRING(X'01')


PiAddr ::= MACAddress

PhysicalDropNumber ::= OCTET STRING (4) -- implementation dependent

PoAddr ::= MACAddress

ProductInstanceId ::= OCTET STRING



ReconParameters ::= SEQUENCE {
```

```
    managementInterlockFlag [0] IMPLICIT BOOLEAN,
    mergePermissionFlag     [1] IMPLICIT BOOLEAN,
    timeTD1A   [4] IMPLICIT OCTET STRING(2),
    timeTD1B   [5] IMPLICIT OCTET STRING(2),
    timeTD2    [6] IMPLICIT OCTET STRING(2),
    timeTD3    [7] IMPLICIT OCTET STRING(2),
    timeTD4    [8] IMPLICIT OCTET STRING(2),
    timeTD5    [9] IMPLICIT OCTET STRING(2),
    timeTD6    [10] IMPLICIT OCTET STRING(2),
    timeTD8    [11] IMPLICIT OCTET STRING(2),
    timeTDA    [12] IMPLICIT OCTET STRING(2) }


 RegisteredList ::= SET OF RegistrationInfo

 RegistrationInfo ::= SEQUENCE {
   groupFunctionTitle [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
   qualifier                        ANY DEFINED BY groupFunctionTitle,
   distinguishingAttribute          ANY DEFINED BY groupFunctionTitle,
   managingProcessInfo              [3] IMPLICIT SET OF ManagingProcessInfo }


-- Resource Type ID as defined in 802.1
   ResourceTypeId ::= SEQUENCE {
                resource [0] IMPLICIT ResourceType,
                revision [1] IMPLICIT StandardRevision,
                options [2] IMPLICIT IEEE802-xLmeOptions,
                manufacturer [3] ManufacturerId OPTIONAL,
                product [4] ManufacturerProductId OPTIONAL,
                version [5] ManufacturerProductVerion OPTIONAL }

-- Non-standard definitions shall use negative numbers
   ResourceType ::= INTEGER {
                ieee8021 (1),
                ieee8022 (2),
                ieee8023 (3),
                ieee8024 (4),
                ieee8025 (5),
                ieee8026 (6),
                ieee8025Bridge (7)    }

 RingStationStatus ::= ANY  -- Implementation dependent

 RingUtilization ::= INTEGER

 Route ::= OCTET STRING


 RW ::= INTEGER

 SAddr ::= MACAddress

 SecurityAccessField ::= OCTET STRING
```

```
SegmentDataRate ::= OCTET STRING(4)

SegmentNumber ::= OCTET STRING

SerialNumber  ::= GraphicString

ServicesSupported ::= LLCServices

StandardRevision ::= INTEGER


StatusType ::= INTEGER {
        notpresent (0),
        active     (1),
        deactivated (2) }

SupportedTypes ::= LLCServices


Timer ::= INTEGER

TopologyInfo  ::= SEQUENCE {
  configErrPresent  [0] IMPLICIT ConfigErrPresent,
  maxAMs            [1] IMPLICIT OCTET STRING(1),
  AMStatus      [2] IMPLICIT AMStatusType
  addrInsert        [3] IMPLICIT AddrInsertType

                                  }
TRLayer1Name ::= OCTET STRING(X'00')


Type3ReceiveResources ::= BOOLEAN

UserData ::= ANY


VendorAdapterCode ::= OCTET STRING(1)
VendorAdapterDescription ::= PRINTABLE STRING(1..40)

WallPlugNumber ::= CHOICE {NULL, GraphicString}




WrapType ::= INTEGER {
        merged (0),
        wrapRI (1),
        wrapRO (2),
        wrapRIRO (3) }

END
```

23.2 LAN-NOTIFIES

```
-- ASN.1 Module for LAN Notifications

LAN-NOTIFIES
 DEFINITIONS ::= BEGIN


-- General Event
-- Note: the GeneralReport data type is conveyed by eventData in
-- EventReportArgument of CMIP


 GeneralReport ::= SEQUENCE {
      GeneralEventDescriptionAndCause,
      GeneralData,
      [11] IMPLICIT LAN-COMMON.LLCStatus OPTIONAL,
      [12] IMPLICIT LAN-COMMON.Correlator OPTIONAL,
      [13] UserData OPTIONAL }

 GeneralEventDescriptionAndCause ::= CHOICE {
      [2] IMPLICIT LSAPPairGeneralEvent,
      [3] IMPLICIT ConcentratorGeneralEvent,
      [4] IMPLICIT GeneralEvent,
     [16] IMPLICIT EthernetGeneralEvent }

EthernetGeneralEvent::= INTEGER {
     OverrunError (1),
     LateCollision (2),
     CarrierLost (3),
     CDHeartbeatError (4),
     UnderrunError (5) }

LSAPGeneralEvent ::= INTEGER {
     LSAPOpened (1) }

LSAPPairGeneralEvent ::= INTEGER {
     linkStationConnected (1),
     linkStationDisconnected (2) }


ConcentratorGeneralEvent ::= INTEGER {
     newComAddress (1),
     lobeStatusChange (2),
     amStatusChange (4),

GeneralEvent ::= INTEGER {
     setOccurred (2),
     nonRegisteredGet (3),
     deviceOnline (4),
     deviceOffline (5) }

GeneralData ::= CHOICE {
```

```
        [6] IMPLICIT DisconnectionData, -- for LSAPPair disconnection
        [5] IMPLICIT LinkConnectionData,
        [7] IMPLICIT ComAddr,
        [8] IMPLICIT LAN-COMMON.LobeStatusEventData,
        [9] IMPLICIT LAN-COMMON.WrapType,
        [10] IMPLICIT LAN-COMMON.AMStatusEventData,
        [11] IMPLICIT LSAPOpened,
        [14] IMPLICIT SetOccurred,
        [15] IMPLICIT NonRegisteredGet }


-- This is the disconnection data if the event is generated as a result
-- of the sublayer being disconnected.

DisconnectionData ::= NULL

-- This is the data if the event is generated as a result
-- of a link connection.

LinkConnectionData ::= NULL

-- This is the data if the event is generated as a result of a
-- LSAP being opened

 LSAPOpened ::= SEQUENCE {
    lsapId   [0] LAN-COMMON.LSAPId,
    supportedTypes              [1] LAN-COMMON.SupportedTypes,
    maxLLCIfield                [2] LAN-COMMON.MaximumLLCInformationFieldSize,
    maximumLinkStnConfigured    [3] LAN-COMMON.MaximumLinkStationsConfigured }

-- This is the data if the event is generated as a result
-- of a new ComAddr.

ComAddr ::= LAN-COMMON.MACAddress
-- This is the data if the event is generated as a result
-- of set from a station that is not a registered managing process.

SetOccurred ::= SEQUENCE {
        [0] IMPLICIT MACAddress,
        [1] IMPLICIT SET OF SEQUENCE {
            attributeId CMIP.AttributeId,
            attributeValue [1] ANY DEFINED BY attributeId OPTIONAL,
            modifyOperator [2] CMIP.ModifyOperator DEFAULT replace } }

-- This is the data if the event is generated as a result
-- of a get received from a station that is not a registered managing
-- process.

NonRegisteredGet ::= SEQUENCE {
        [0] IMPLICIT MACAddress,
        attributeList [1] IMPLICIT SET OF CMIP.Attribute }


-- Note: The Function Present data type is conveyed by eventData in
-- EventReportArgument of CMIP
 FunctionPresent ::= SEQUENCE {
```

```
    groupFunctionTitle [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
    qualifier           ANY DEFINED BY groupFunctionTitle,
    distinguishingAttribute  ANY DEFINED BY groupFunctionTitle,
    primaryName       [3] IMPLICIT DARRD.RegularIdentifier,
    macAddress        [4] IMPLICIT LAN-COMMON.MACAddress,
    archRelLevel      [6] IMPLICIT LAN-COMMON.ArchReleaseLevel,
    userdata           ANY DEFINED BY groupFunctionTitle OPTIONAL
                                        }

-- Note: The Deregister Event data type is conveyed by eventData in
-- EventReportArgument of CMIP

  Deregister ::= SEQUENCE {
    groupFunctionTitle [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
    qualifier           ANY DEFINED BY groupFunctionTitle,
    distinguishingAttribute  ANY DEFINED BY groupFunctionTitle,
    primaryName       [3] IMPLICIT DARRD.RegularIdentifier,
    macAddress        [4] IMPLICIT LAN-COMMON.MACAddress,
    userdata           ANY DEFINED BY groupFunctionTitle
                                        }


-- Note: The Multiple Function Present data type is conveyed by
-- eventData in EventReportArgument of CMIP

  MultipleFunctionPresent ::= SEQUENCE {
     SET OF { SEQUENCE {
        groupFunctionTitle [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
        qualifier         ANY DEFINED BY groupFunctionTitle,
        distinguishingAttribute  ANY DEFINED BY groupFunctionTitle,
        userdata          ANY DEFINED BY groupFunctionTitle } }

     primaryName      [3] IMPLICIT DARRD.RegularIdentifier,
     macAddress       [4] IMPLICIT LAN-COMMON.MACAddress,
     archRelLevel     [6] IMPLICIT LAN-COMMON.ArchReleaseLevel
      }

-- Note: The Deregister Event data type is conveyed by eventData in
-- EventReportArgument of CMIP

  MultipleFunctionDeregister ::= SEQUENCE {
     SET OF {  SEQUENCE {
        groupFunctionTitle [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
        qualifier         ANY DEFINED BY groupFunctionTitle,
        distinguishingAttribute  ANY DEFINED BY groupFunctionTitle,
        userdata          ANY DEFINED BY groupFunctionTitle } }
     primaryName      [3] IMPLICIT DARRD.RegularIdentifier,
     macAddress       [4] IMPLICIT LAN-COMMON.MACAddress
                                        }


  END
```

23.3 LAN-ACTIONS

```
LAN-ACTIONS
 DEFINITIONS ::= BEGIN

 SAPData ::= LAN-COMMON.LLCServices

 ReinitializeData ::= NULL

 CorrelatorData ::= LSAPPairId

 CorrelatorRspData ::= LAN-COMMON.Correlator


 RegisterReqData ::= SEQUENCE {
     groupFunctionTitle      [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
     qualifier               ANY DEFINED BY groupFunctionTitle,
     distinguishingAttribute  ANY DEFINED BY groupFunctionTitle,
     primaryName             [3] IMPLICIT DARRD.RegularIdentifier,
     macAddress              [4] IMPLICIT LAN-COMMON.MACAddress,
     mpName                  [5] IMPLICIT DARRD.RegularIdentifier,
     mpAddress               [6] IMPLICIT LAN-COMMON.MACAddress,
     mpLevel                 [7] IMPLICIT LAN-COMMON.ArchReleaseLevel,
     lostMPRetries           [8] IMPLICIT Integer,
     securityAccessField     [9] IMPLICIT LAN-COMMON.SecurityAccessField,
     mpSAP                   [10] IMPLICIT LAN-COMMON.LSAP,
     cnfEventRetryTimeout    [11] IMPLICIT LAN-COMMON.Timer OPTIONAL,
     cnfEventRetryNumber     [12] IMPLICIT INTEGER OPTIONAL,
     userData                ANY DEFINED BY groupFunctionTitle OPTIONAL {


RegisterReqRspData ::= SEQUENCE {
     groupFunctionTitle      [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
     qualifier               ANY DEFINED BY groupFunctionTitle,
     distinguishingAttribute  ANY DEFINED BY groupFunctionTitle,
     primaryName             [3] IMPLICIT DARRD.RegularIdentifier,
     macAddress              [4] IMPLICIT LAN-COMMON.MACAddress,
     archReleaseLevel        [6] IMPLICIT LAN-COMMON.ArchReleaseLevel,
     returnCode              [7] RegisterReturnCode,
     securityAccessField     [8] IMPLICIT LAN-COMMON.SecurityAccessField,
     characterSet            [9] IMPLICIT CharacterSet OPTIONAL,
       -- default if not present is ISO 8859-1, Latin Alphabet No. 1
     userData                 ANY DEFINED BY groupFunctionTitle OPTIONAL }

CharacterSet ::= OCTETSTRING

DeregisterReqData ::= SEQUENCE {
 groupFunctionTitle          [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
  qualifier                   ANY DEFINED BY groupFunctionTitle,
  distinguishingAttribute    ANY DEFINED BY groupFunctionTitle,
  primaryName                [3] IMPLICIT DARRD.RegularIdentifier,
  macAddress                 [4] IMPLICIT LAN-COMMON.MACAddress,
  mpName                     [5] IMPLICIT DARRD.RegularIdentifier,
```

```
    mpAddress               [6] IMPLICIT LAN-COMMON.MACAddress,
    userData                 ANY DEFINED BY groupFunctionTitle OPTIONAL}


  RegisterCheckData ::=  SEQUENCE {
      mpName                [0] IMPLICIT DARRD.RegularIdentifier,
      mpAddress             [1] IMPLICIT LAN-COMMON.MACAddress,
      groupFunctionTitle    [2] IMPLICIT LAN-COMMON.GroupFunctionTitle,
      qualifier              ANY DEFINED BY groupFunctionTitle,
      mpSAP                 [3] IMPLICIT LAN-COMMON.LSAP,
      responseType          [4] IMPLICIT ResponseType }


  RegisterCheckRspData ::=  SEQUENCE {
     groupFunctionTitle     [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
     qualifier              ANY DEFINED BY groupFunctionTitle,
     distinguishingAttribute ANY DEFINED BY groupFunctionTitle,
     primaryName            [3] IMPLICIT DARRD.RegularIdentifier,
     macAddress             [4] IMPLICIT LAN-COMMON.MACAddress,
     thisMPRegistered       [5] IMPLICIT BOOLEAN,
     registerList           [6] IMPLICIT LAN-COMMON.RegisteredList }

RemoveStationData ::=  SEQUENCE {
 [0] IMPLICIT LAN-COMMON.MacAddress,
 securityAcc [1] IMPLICIT LAN-COMMON.SecurityAccessField }

 RemoveStationRspData ::=  SEQUENCE {
  [0] IMPLICIT RemoveStationReturnCode   }

 WrapData ::= LAN-Common.WrapType

 SoftResetData ::= NULL

 RPUEnableData ::= SEQUENCE {
       loadFile [0] IMPLICIT OCTET STRING(1..24),
       loaderAddress [1] IMPLICIT MacAddress OPTIONAL
                                          }

 RegisterReturnCode ::= INTEGER  {
                 registered (0),
                 reregistered (1),
                 groupFunctionTitleNotPresent (2),
                 regRejectedTooManyManagers(3),
                 accessDenied (4),
                 qualifierDoesNotMatch (5) }


 RemoveStationReturnCode ::= INTEGER  {
                 removed   (0),
                 ignored   (1) }

ResponseType ::= INTEGER {
    rspIfGroupFunctionAndQualifierRegistered (0),
    rspIfGroupFunctionAndQualifierNotRegistered (1),
    rspIfGroupFunctionAndQualifierExists (2),  -- registered or not
    rspIfGroupFunctionRegister (4),            -- ignore qualifier
```

```
    rspIfGroupFunctionNotRegistered (5),      -- ignore qualifier
    rspIfGroupFunctionExist (6)               -- ignore qualifier
                              }

MultipleRegisterReqData ::= SEQUENCE {
  SET OF { SEQUENCE OF {
     groupFunctionTitle      [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
     qualifier               ANY DEFINED BY groupFunctionTitle,
     distinguishingAttribute ANY DEFINED BY groupFunctionTitle,
     lostMPRetries           [8] IMPLICIT Integer,
     securityAccessField     [9] IMPLICIT LAN-COMMON.SecurityAccessField,
    cnfEventRetryTimeout     [11] IMPLICIT LAN-COMMON.Timer OPTIONAL,
    cnfEventRetryNumber      [12] IMPLICIT INTEGER OPTIONAL,
     userData                ANY DEFINED BY groupFunction OPTIONAL }}
  primaryName                [3] IMPLICIT DARRD.RegularIdentifier,
  macAddress                 [4] IMPLICIT LAN-COMMON.MACAddress,
  mpName                     [5] IMPLICIT DARRD.RegularIdentifier,
  mpAddress                  [6] IMPLICIT LAN-COMMON.MACAddress,
  mpLevel                    [7] IMPLICIT LAN-COMMON.ArchReleaseLevel,
  mpSAP                      [10] IMPLICIT LAN-COMMON.LSAP }


MultipleRegisterReqRspData ::= SEQUENCE {
  SET OF { SEQUENCE OF {
     groupFunctionTitle      [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
     qualifier               ANY DEFINED BY groupFunctionTitle,
     distinguishingAttribute ANY DEFINED BY groupFunctionTitle
     returnCode              [7] RegisterReturnCode,
     securityAccessField     [8] IMPLICIT LAN-COMMON.SecurityAccessField
     userData                ANY DEFINED BY groupFunctionTitle OPTIONAL }
                                  }
  primaryName                [3] IMPLICIT DARRD.RegularIdentifier,
  macAddress                 [4] IMPLICIT LAN-COMMON.MACAddress,
  archReleaseLevel           [6] IMPLICIT LAN-COMMON.ArchReleaseLevel
                                  }

MultipleDeregisterReqData ::= SEQUENCE {
 SET OF {
   SEQUENCE OF {
    groupFunctionTitle       [0] IMPLICIT LAN-COMMON.GroupFunctionTitle,
    qualifier                ANY DEFINED BY groupFunctionTitle,
    distinguishingAttribute  ANY DEFINED BY groupFunctionTitle,
     characterSet            [9] IMPLICIT CharacterSet OPTIONAL,
       -- default if not present is ISO 8859-1, Latin Alphabet No. 1
    userData                 ANY DEFINED BY groupFunctionTitle OPTIONAL}
  primaryName                [3] IMPLICIT DARRD.RegularIdentifier,
  macAddress                 [4] IMPLICIT LAN-COMMON.MACAddress,
  mpName                     [5] IMPLICIT DARRD.RegularIdentifier,
  mpAddress                  [6] IMPLICIT LAN-COMMON.MACAddress }
END
```

23.4 Layer Counters

```
MSOAS-LAYER-COUNTERS
 DEFINITIONS ::= BEGIN

IMPORTS ObjectClass, ObjectInstance FROM
      CMIP-1 {joint-iso-ccitt ms(9) cmip(1) version(1) protocol(3)}
      ManagerPointer FROM MSOAS-SUPPORT-OBJECTS;


-- This attribute identifies the class of the object instance
-- under which the ThresholdControl managed object emitting the
-- CounterSetReport is contained.  It is thus the class to which
-- the counters reported in the CounterSetReport apply.
ClassReportedOn ::= ObjectClass


CounterData ::= SEQUENCE {
      counterName    [0] IMPLICIT CounterName,
      currentValue   [1] IMPLICIT CounterValue,
      intervalStartValue   [2] IMPLICIT CounterValue
      }


CounterName ::= OCTET STRING SIZE (2)


CounterSetData ::= SET OF CounterData


CounterSetReply ::= SET OF SEQUENCE {
      counterName     [0] IMPLICIT CounterName,
      currentValue    [1] IMPLICIT CounterValue
      }


CounterSetReport ::= CHOICE {
      instanceDeletion  [0] IMPLICIT CounterSetReportCommonData,
      containingInstanceDeletion  [1] IMPLICIT CounterSetReportCommonData,
      counterWrap  [2] IMPLICIT SEQUENCE {
                  [0] IMPLICIT CounterSetReportCommonData,
                  [1] IMPLICIT CounterWrapData
                  },
      thresholdReached  [3] IMPLICIT SEQUENCE {
                  [0]  IMPLICIT  CounterSetReportCommonData,
            numeratorData  [1]  IMPLICIT  TriggerData,
            denominatorData  [2]  IMPLICIT  TriggerData OPTIONAL
            }
      }


CounterSetReportCommonData ::= SEQUENCE {
      classReportedOn    [0] IMPLICIT ClassReportedOn,
      managerPointer   [1] IMPLICIT ManagerPointer,
```

```
        counterSetData  [2] IMPLICIT CounterSetData
        }

CounterValue ::= INTEGER

-- The following is a SET OF because multiple counter wraps
-- may be reported in a single CounterSetReport
CounterWrapData ::= SET OF SEQUENCE {
        counterName     [0] IMPLICIT CounterName,
        counterMaxValue [1] IMPLICIT CounterValue
        }

-- The following integer represents an exponent of 10, with
-- units of seconds. E.g., -1 = 10**(-1) seconds = 100 milliseconds.
MaxSampleInterval ::= INTEGER

Offset ::= INTEGER

-- This attribute identifies the class of the instances
-- under which ThresholdControl managed objects implementing
-- the specified policy are to be created.
PolicyTargetScope ::= ObjectClass

ReportSwitch ::= BOOLEAN

RequiredMaxSampleInterval ::= MaxSampleInterval (-2..1)

ThresholdControlInitialValuesName ::= PrintableString

ThresholdControlName ::= PrintableString

ThresholdPolicyPointer ::= CHOICE {
        NULL,
        ObjectInstance
        }

ThresholdPolicyValuesName ::= PrintableString

ThresholdSubtreePolicyName ::= PrintableString

-- Exactly one ThresholdSubtreePolicy instance exists under each
-- non-leaf layer instance, so these instances can all have the
-- the same RDN.  Suggestions are solicited for a more NLS-friendly
```

```
-- way of accomplishing this.
ThresholdSubtreePolicyNameValue ::=
        ThresholdSubtreePolicyName ("TSUBTREEPOLICY")


Trigger ::= SEQUENCE {
    numeratorCName      [0] IMPLICIT CounterName,
    numeratorOffset     [1] IMPLICIT Offset,
    denominatorCName    [2] IMPLICIT CounterName OPTIONAL,
    denominatorOffset   [3] IMPLICIT Offset OPTIONAL
    }

TriggerData ::= SEQUENCE {
    triggerCName      [0]  IMPLICIT CounterName,
    triggerOffset     [1]  IMPLICIT Offset,
    triggerValueAtEffectiveReset    [2]  IMPLICIT CounterValue,
    triggerCounterMaxValue    [3]  IMPLICIT CounterValue
    }


TriggerList ::= SET OF Trigger


END
```

23.5 Support

```
MSOAS-SUPPORT-OBJECTS
 DEFINITIONS ::= BEGIN
LibraryObjectName ::= PrintableString


-- Exactly one LibraryObject instance exists in a managed
-- system, directly under the object named by SystemTitle.
-- Suggestions are solicited for a more NLS-friendly way of naming
-- the LibraryObject managed object.
LibraryObjectNameValue ::=
        LibraryObjectName ("LIBRARY-OBJECT")

-- The syntax for ManagerPointer, as well as the template for the
-- attribute that stores the value in an object, will be defined in
-- conjunction with the MS Framework (AWP-298).
ManagerPointer ::= -- TBD
END
```

Chapter 24. Group Function Title Table

This table defines the group function title values which are sent in the Registration flows. The first column specifies the function type. The second column gives the value which flows in the group function title field. The third column defines what attribute is the management qualifier for the group function title. The fourth column defines the managed object classes which make up the function type.

Function Type	Group Function Title	Qualifier	Distinguishing Attribute	Managed Object Classes
Token-Ring MAC And LLC	X'01 00'	Segment Number	MAC Address	Resource Management Token-Ring Layer 1 Token-Ring Layer 2 - MAC Environment LAN Layer 2 - LLC LSAP LSAP Pair Counter Threshold Control Threshold Subtree
Minimal Token-Ring	X'01 01'	Segment Number	MAC Address	Resource Management Token-Ring Layer 1 Token-Ring Layer 2 - MAC Environment
CAU	X'01 02'	Segment Number	Access Unit Id	CAU Attachment Module
DAR/RD Name Management	X'01 03'	Segment Number	MAC Address	Name Management
PCNet	X'01 04'	Segment Number	MAC Address	• PC Network Layer 1 • PC Network Layer2
Ethernet	X'01 05'	Segment Number	MAC Address	• Ethernet Layer 1 • Ethernet Layer 2 • Resource Management
Ethernet and LLC	X'01 05'	Segment Number	MAC Adress	• Ethernet Layer 1 • Ethernet MAC Layer Environment LAN Layer 2 - LLC LSAP LSAP Pair Counter Threshold Pair
Ethernet and Minimum LLC	X'01 06'	Segment Number	MAC Adress	• Ethernet Layer 1 • Ethernet MAC Layer Environment Counter Threshold Pair *LAN Layer 2 - LLC *LSAP *LSAP Pair
Minimum Ethernet	X'01 07'	Segment Number	MAC Adress	• Ethernet Layer 1 • Ethernet MAC Layer Environment Counter

* **Note**: This Managed Object support is minimum for this Function Group, that is, its LLC is related to station manager only.

Appendix A. References

A.1.1 External IBM publications

SC30-3346 IBM Systems Network Architecture Management Services Reference

SC30-3374-02 IBM Token-Ring Network Architecture Reference

A.1.2 OSI Standards References

10165-2 Definition of Management Attributes

ISO/IEC 7498, Information processing systems - Open Systems Interconnection - Basic Reference Model

ISO/IEC 7498-4, Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 4

ISO/IEC 8824, Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)

ISO/IEC 8825, Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)

ISO/IEC 9072-1, Information processing systems - Open Systems Interconnection - Remote Operations - Part 1: Model, Notation and Service Definition

ISO/IEC 9072-2, Information processing systems - Open Systems Interconnection - Remote Operations - Part 2: Protocol Specification.

ISO/IEC 9595, Information processing systems - Open Systems Interconnection - Common Management Information Service Definition.

ISO/IEC 9596, Information processing systems - Open Systems Interconnection - Common Management Information Protocol Specification.

ISO/IEC 9596, Information processing systems - Open Systems Interconnection - Common Management Information Protocol Specification.

ISO/IEC 2nd DP10040, Information processing systems - Open Systems Interconnection - Systems Management Overview

ISO/IEC 2nd DP10040, Information processing systems - Open Systems Interconnection - Systems Management Overview

ISO/IEC DP 10164-5, Information processing systems - Open Systems Interconnection - Systems-Management - Part 5: Event Report Management Function

ISO/IEC DP 10165-4, Information processing systems - Open Systems Interconnection - Management Information Services - Structure of Management Information Part 4: Guidelines for the Defintiions of Managed Objects.

Appendix B. Operating System Considerations

As previously noted, this architecture is intended to be operating system independent. Therefore, the formats and protocols described in this document, are not wedded to the transport requirements of any one operating system. For implementation examples, see the accompanying HLM documentation; *Heterogeneous LAN Management: Technical Reference*. These examples represent different applications of this architecture on two different, OS specific, transport mechanisms.

Appendix C. Media Considerations

C.1 Ethernet

Several design considerations are factored in HLM architecture to ensure Ethernet inter-operability. The base inter-operability between Ethernet and Token-Ring is assured by proper specification of the "group" address. These group addresses provide a set of well known MAC IDs to support the functions such as Discovery and Heartbeating. As expressed in 802.3 terms, these addresses are MULTICAST addresses that exactly map to the corresponding 802.5 "functional" addresses. This mapping of addresses provides a common address base for all 802 media. MAC level bridges will forward on these PDUs across media boundaries and are in a proper format for both Ethernet and Token-Ring adapters to receive.

Group addresses, as described in HLM, have a multicast format that is different than the customary format and are also presented in canonical format. In canonical order, the bits are presented in the order which they are transmitted on the LAN media and NOT how they are presented in memory of the host CPU.

Additionally, assignment of multicase addresses typically by vendor ID where the multicast addresses are created by setting the appropriate broadcast bits in the vendor ID field. With HLM "group addresses" do not conform to this convention and have no vendor ID as part of the multicast addresses.

To inter-operate in a MAC bridge environment, all Ethernet implementations of HLM must conform to the 802.3 frame format specifications (as opposed to DIX Ethernet or SNAPped frames). This requirement will ensure inter-vendor Ethernet inter-operability.

C.2 Token-Ring Considerations

C.2.1 Functional Addresses

802.5 LANs provide bit-specific *functional addresses* for widely used functions, such as the configuration report server. Ring stations use functional address "masks" to identify these functions.

For example, if function G is assinged a functional address of X'C000 0008 0000', and function M is assigned a functional address of X'C000 0000 0040', then ring station Y, whose node contains functions G and M, would have a mask of X'C000 0008 0040'.

All functional addresses are locally administered group addresses (bits 0 and 1 of byte 0 of the destination address are set to B'11'). Bit 0 of byte 2 of the destination address (the functional address indicator) is set to B'0' for functional addressses. The remaining 7 bits in byte 2, and the 8 bits each in bytes 2, 3, and 5 allow up to 31 functional addresses to be defined (because the addresses are bit-specific).

The functional addresses listed below have been defined pertaining to functions defined in this document:

Function Name	Functional Address
LAN Manager	X'C000 0000 2000'
Resource Manager	X'C000 0002 0000'
Discovery Heartbeat	X'C000 0000 0004'
Discovery Non-Server	X'C000 0000 0040'
Discovery Server	X'C000 0001 0000'

C.2.2 Bit Ordering of Addresses in Token-Ring LANs

All addresses carried as LLC Data shall be transmitted using canonical bit ordering. Canonical bit ordering means that the least significant bit in every byte is transmitted first, as opposed to the most significant bit first that is transmitted in the source and destingation addresses in that Token-Ring MAC header.

Addresses in event data, object instance names, action data, SETs and GETs shall be transmitted using canonical bit ordering.

Appendix D. Abstract Syntax Notation (ASN.1)

The contents of this appendix is taken from the following two Standards documents:

1. ISO 8824 - Specification of Abstract Syntax Notation One (ASN.1)
2. ISO 8825 - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)

The CMIP and ROS parameters are encoded based on the ASN.1 recommendation. The following section briefly describes the encoding mechanism.

Each element as defined in the ASN.1 has three components as shown:

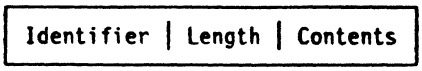
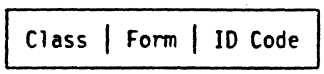


Figure D-1. General ASN.1 Data Structure

1. Identifier - at least 1 octet and is broken down into three parts as shown:




```
Class : 00 = universal
        01 = application
        10 = context specific
        11 = private use

Form  :  0 = primitive
         1 = constructor

ID code  : 0-30, specific ID code value
           31, extension octet(s) follow, bit 8 of the last extension
               is 0, ID code is the concatenation of bits 7-1.

               For each extended octet:
                 Bit(8) means: last octet if '0' or extended if '1'
                 Bit(1..7) concatenated value express the ID code value

               End of contents is specified by X'0000'.
```

Figure D-2. Identifier Portion of the ASN.1 Data Structure

As shown, there are four classes, Universal class is only used as specified within this international standard. They are defined in Figure D-3. Application class is assigned by other standards. Context specific class is freely assigned within any use of this notation, and is interpreted according to the context in which it is used. Private class is never assigned by international standards, its use is enterprise specific.

Two forms of data elements are distinguished: A primitive element is one whose contents is atomic, i.e., has no further internal structure of data element. A constructor is one whose contents is itself a data element, or a series of data elements. Constructor elements are thus recursively defined.

If the class is specified as universal, then the ID code values are defined as follows:

Universal 1	Boolean
Universal 2	Integer
Universal 3	Bit string
Universal 4	Octet string
Universal 5	Null
Universal 6	Object identifier
Universal 7	Object descriptor
Universal 8	External
Universal 9–15	reserved
Universal 16	Sequence
Universal 17	Set
Universal 18–22	Character string
Universal 23	UTC time
Universal 24	Generalized time
Universal 25–27	Character string
Universal 28–...	reserved

Figure D-3. Universal ID Code Assignment

- Boolean - A type with two distinguished values: true or false.
- Integer - A type with distinguished values which are the positive and negative whole numbers.
- Bit string - A type whose distinguished values are an ordered sequence of zero, one or more bits.
- Octet string - A type whose distinguished values are an ordered sequence of zero, one or more octets. Each octet being an ordered sequence of eight bits.
- Null - A type consisting of a single value called null or empty string.
- Object identifier - A type whose distinguishable values is associated with an information object.
- Object descriptor - A type whose distinguished values are human-readable text providing a brief description of an information object.
- External - A type whose distinguished values cannot be deduced from their characterization as external, but which can be deduced from the encoding of such a value; the values may not conform to ASN.1 encoding rules.
- Sequence - A fixed, ordered list of types: each value in the new type is an ordered list of values, one from each of the component type.
- Set - A fixed, unordered list of distinct types: each value in the new type is an unordered list of values, one from each of the component type.
- Character string - A type whose values are strings of characters from some defined character set.
- UTC time - A particular form of generalized time which is defined for use in international application where the local time only is not adequate.
- Generalized time - represents a calendar date and time of day as provided by ISO 2014, ISO3307, and ISO 4031. The time of day can be specified as local time only, UTC time only, or as both local and UTC time.

2. Length field - indicates the number of the octets for the contents. It has three forms as shown:

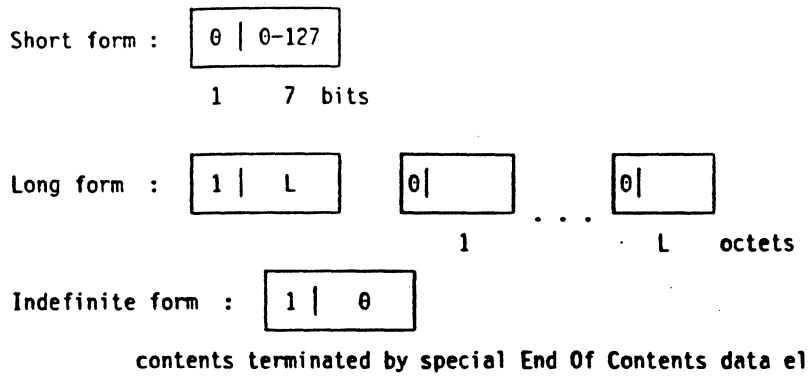
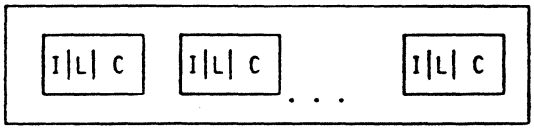


Figure D-4. Length Portion of the ASN.1 Data Structure

If the content field is less than 127 octets, short form is used. If the content is greater than 127 octets, then long form is used.

3. Contents - If primitive is chosen for the form, then the contents will just be the data, no more further breakdown is needed. However, if constructor is chosen for the form, the contents will consists of the following sequence:



I = Identifier, which has the same format as shown in Figure 15.
L = Length, which has the same format as shown in Figure 16.
C = Contents, which can be further divided into I, L, and C again.

Figure D-5. Contents Portion of the ASN.1 Data Structure

Appendix E. Remote Operations (ROS)

The contents of this appendix is taken from the following two Standards documents:

1. DIS 9072/1, Remote Operations: Model, Notation and Service Definition
2. DIS 9072/2, Remote Operations: Protocol Specification

ROS provides a vehicle for the communication between two management application entities. The following services are provided based on the Standards recommendation:

1. RO-Invoke - request that an operation to be performed. It has the following parameters:
   a. Invoke ID - This required parameter identifies the request and is used to correlate this request with the corresponding replies.
   b. Linked ID - This optional parameter is used to link operations which is performed by one parent operation and one or more child operations. The performed of the parent operation may invoke none, one, or more child operations during the execution of the parent operation. If this parameter is present, the invoked operation is a child operation and the parameter identifies the invocation of the linked parent operation.
   c. Operation Code - This required parameter is the identifier of the operation to be invoked. This value has to be agreed between the ROS users.
   d. Argument - This optional parameter is the argument of the invoked operation. Again, the contents must be agreed between the ROS users.

   Figure E-1 summarize the RO-Invoke service and its parameters:

parameter Name	Request
Invoke ID	M
Linked ID	U
Operation code	M
Arguments	U

M - Mandatory
U - Un-mandatory, optional

Figure E-1. RO Invoke Parameters

There are two sequence of service primitives associated with this service:

- RO-INVOKE.req
- RO-INVOKE.ind

The sequence in which these primitives are used is illustrated below:

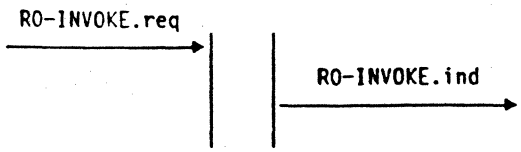


Figure E-2. RO Invoke Service Primitives

2. RO-Result - reports the successful completion of an operation. It has the following parameters:

   a. Invoke ID - This required parameter identifies the request and is used to correlate this request with the corresponding replies.

   b. Argument - this optional parameter is the result of an invoked and successfully performed operation. The contents has to be agreed between the ROS users.

   Figure E-3 summarize the RO-Result service and its parameters:

parameter Name	Request
Invoke ID	M
Arguments	U

M – Mandatory
U – Un-mandatory, optional

Figure E-3. RO Result Parameters

There are two sequence of service primitives associated with this service:

- RO-RESULT.req
- RO-RESULT.ind

The sequence in which these primitives are used is illustrated below:

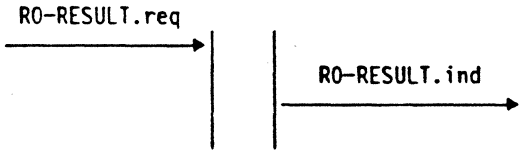


Figure E-4. RO Result Service Primitives

3. RO-Error - reports the unsuccessful completion of an operation. It has the following parameters:

   a. Invoke ID - This required parameter identifies the request and is used to correlate this request with the corresponding replies.

   b. Error Code - This required parameter identifies the error occurred during the operation. the contents of the error code must be agreed between the ROS users.

   c. Parameter - Additional information regarding the error code, optional.

   Figure E-5 summarize the RO-Error service and its parameters:

parameter Name	Request
Invoke ID	M
Error Code	M
Arguments	U

M - Mandatory
U - Un-mandatory, optional

Figure E-5. RO Error Parameters

There are two sequence of service primitives associated with this service:

- RO-ERROR.req
- RO-ERROR.ind

The sequence in which these primitives are used is illustrated below:

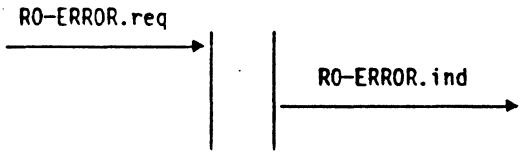


Figure E-6. RO Error Service Primitives

4. RO-Reject-User - The user-reject is used by one application entity to reject the request or reply of the other application entity. It has the following parameters:

   a. Invoke ID - This required parameter identifies the request and is used to correlate this request with the corresponding replies.

   b. Problem Code - This parameter identifies the error occurred during the invoke operation. The problem code are defined as follows:

      1) Duplicate invocation - signifies that the Invoke ID parameter violates the assignment rules.
      2) Unrecognized operation - signifies that the operation is not one of those agreed between the users.
      3) mistyped argument - signifies that the type of the operation argument supplied is not agreed between the users.
      4) Resource limitation - the performing ROS user is not able to perform the invoke operation due to resource limitation.
      5) Initiator releasing - the association initiator is not willing to perform the invoked operation because it is about to attempt to release the application association.
      6) Unrecognized linked ID - signifies that there is no operation in progress with an invoked ID equal to the specified linked ID.
      7) Linked response unexpected - signifies that the invoked operation referred to by the linked ID is not a parent operation.
      8) Unexpected child operation - signifies that the invoked child operation is not one that the invoked parent operation referred to by the linked ID allows.

Figure E-7 summarize the RO-Error service and its parameters:

parameter Name	Request
Invoke ID	M
Problem code	M

M - Mandatory
U - Un-mandatory, optional

Figure E-7. RO Reject User Parameters

There are two sequence of service primitives associated with this service:

- RO-REJECT.req
- RO-REJECT.ind

The sequence in which these primitives are used is illustrated below:

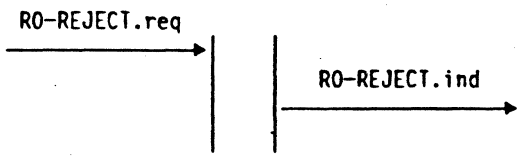


Figure E-8. RO Reject Service Primitives

Appendix F. Common Management Information Protocol (CMIP)

The contents of this appendix is taken from the following two Standards documents:

1. ISO/TC 97/SC 21N 1373, Common management Information Service Definition
2. ISO/TC 97/SC 21N 1375, Common management Information Protocol Definition

CMIP provides request/response service between two system management application entities (SMAE). It also provides an unsolicited event reporting service between an event initiator SMAE and a collector SMAE. The protocol specified supports the following Common Management Information Services (CMIS):

1. Get - This is a service element used by a SMAE to request transfer of management information from another SMAE. It has the following parameters:
   a. Invoke ID - This optional field specifies the identifier assigned to the operation. It can be used to distinguish the present operation from others that the invoking SMAE may have in progress at the invoked SMAE.
   b. Resource ID - This required field identifies the relevant resource. It is the resource to which the management information pertains.
   c. Access Control - This optional field is information of unspecified form to be used as input to access control function in the initiator SMAE and/or responder SMAE on any type of exchange.
   d. Synchronization - This required field indicates how the initiator SMAE wants the responder SMAE to synchronize several operations that are within a PDU to be processed. There are several ways that a series of operations specified in a single operations specified in a single service element may be processed:
      1) Best effort - All the ones that can be done do get done, the ones that can't, don't. The ordering is not important.
      2) Ordered - All the ones that can be done do get done, the ones that can't, don't Operations are attempted in the sequence they are presented in the PDU.
      3) Stop on Error - The operations are processed sequentially in the order they appear in the PDU, and when an unsuccessful operation is encountered, the remaining operations are not attempted.
      4) Atomic - All operations are checked to see if they can be performed, and either all are performed successfully, or none are performed.

      If this field is not specified in the protocol, a default of best effort is to be assumed.
   e. Attribute ID List - This required field contains a set of attribute identifiers that are to be used by the responder SMAE to read their values and return them to the initiator SMAE.

   The response to Get contains one of the following parameters:
   a. Invoke ID - optional, previously defined.
   b. Attribute List - This required field contains a set of attribute identifier/value pairs

c. Status - If error is encountered during the operation, this required field contains the following error codes:

   1) No Such Operation - The operation is not supported.

   2) No Such Resource - The resource ID specified is not recognized.

   3) Access denied - The attributes were not read due to the fact that the value of the access control was not acceptable or access was denied for other reasons.

   4) Synchronization not Supported - The type of synchronization specified in the synchronization parameter is not supported.

   5) No Such Attribute - The attribute specified is not supported

   6) Get List Error - One or more attributes were not read, the attribute values that could be read are returned.

d. Current Time - This optional field contains the time the response was generated. It may be expressed in Generalized time(ISO 3307), UTC time (ASN.1) or local time (implementation defined).

Figure F-1 summarize the Get service and its parameters:

parameter Name	Request	Response
Invoke ID	U	U
Resource ID	M	-
Access Control	U	-
Synchronization	M	-
Attribute ID list	M	-
Current Time	-	U
Attribute list	-	M
Status	-	M

M - Mandatory
U - Un-mandatory, optional

Figure F-1. Get Parameters

There are four sequence of service primitives associated with this service:

- M-GET.req
- M-GET.ind
- M-GET.rsp
- M-GET.conf

The sequence in which these primitives are used is illustrated below:

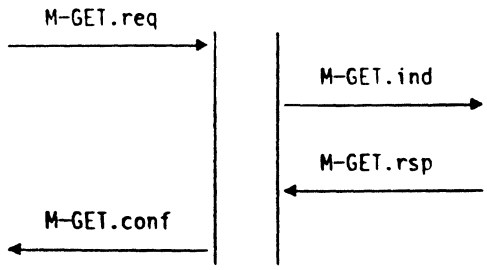


Figure F-2. Get Service Primitives

2. Set - This is a service element used by a SMAE to request another SMAE to set the values of attributes. It has the following parameters:
   a. Invoke ID - optional, previously defined.
   b. Resource ID - required, previously defined.
   c. Access control - optional, previously defined.
   d. Synchronization - required, previously defined.
   e. Attribute List - This required field contains a set of attribute identifier/value pairs that are to be set by the responder SMAE.

   The response to Set contains one of the following parameters:
   a. Invoke ID - optional, previously defined.
   b. Current Time - optional, previously defined.
   c. Status - If error is encountered during the operation, this required field contains the following error codes:
      1) No Such Operation - previously defined
      2) No Such Resource - previously defined
      3) Access denied - previously defined
      4) Synchronization not Supported - previously defined
      5) No Such Attribute - previously defined
      6) Invalid Attribute Value - previously defined
      7) Set List Error - One or more attributes were not set, the attribute values that could be set are indicated.

   Figure F-3 summarize the Set service and its parameters:

parameter Name	Request	Response
Invoke ID	U	U
Resource ID	M	-
Access Control	U	-
Synchronization	M	-
Attribute list	M	U
Current Time	-	U
Status	-	M

M - Mandatory
U - Un-mandatory, optional

Figure F-3. Set Parameters

There are four sequence of service primitives associated with this service:

- M-SET.req
- M-SET.ind
- M-SET.rsp
- M-SET.conf

The sequence in which these primitives are used is illustrated below:

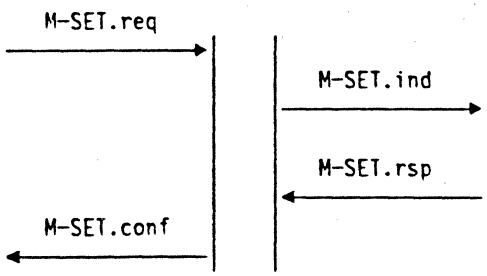


Figure F-4. Set Service Primitives

3. Action - This is a service element used by a SMAE to request another SMAE to perform an operation. It has the following required parameters:

   a. Invoke ID - optional, previously defined.

   b. Resource ID - required, previously defined.

   c. Access control - optional, previously defined.

   d. Action Type - This field specifies a particular action that is to be performed.

   The response to Action contains one of the following parameters:

   a. Invoke ID - optional, previously defined.

   b. Current Time - optional, previously defined.

   c. Action Report - If the operation is successful, this required field returns the result of the successful action performed.

d. Status - If error is encountered during the operation, this required field contains the following error codes:

1) No Such Operation - previously defined

2) No Such Resource - previously defined

3) Access denied - previously defined

4) No Such Action - The action type specified is not supported.

5) No Such Action Argument - One or more of the specified action arguments are not supported.

Figure F-5 summarize the Action service and its parameters:

parameter Name	Request	Response
Invoke ID	U	U
Resource ID	M	-
Access Control	U	-
Action Type	M	-
Current Time	-	U
Action Report	-	M
Status	-	M

M - Mandatory
U - Un-mandatory, optional

Figure F-5. Action Parameters

There are four sequence of service primitives associated with this service:

- M-ACTION.req
- M-ACTION.ind
- M-ACTION.rsp
- M-ACTION.conf

The sequence in which these primitives are used is illustrated below:

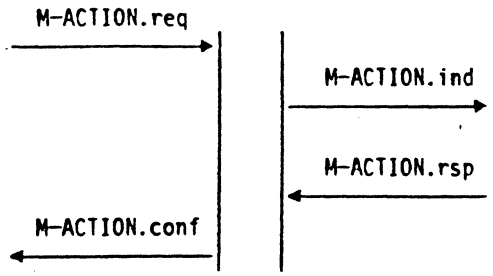


Figure F-6. Action Service Primitives

4. Event Report - This is a service element used by a SMAE to report some unsolicited events of a resource to another SMAE. It has the following parameters:

a. Invoke ID - optional, previously defined.

b. Resource ID - required, previously defined.

c. Event Type - This required field specifies the type of event being reported.

d. Event Time - This required field contains the time of generation of the event.

e. Event Info - This required field identifies information that the initiating SMAE is able to supply about the event.

Since this is an unconfirmed event report, no response is needed.

Figure F-7 summarize the Event Report service and its parameters:

parameter Name	Request	Response
Invoke ID	U	-
Resource ID	M	-
Event Type	M	-
Event Time	M	-
Event Info	M	-

M - Mandatory
U - Un-mandatory, optional

Figure F-7. Event Report Parameters

There are two sequence of service primitives associated with this service:

- M-EVENT-REPORT.req
- M-EVENT-REPORT.ind

The sequence in which these primitives are used is illustrated below:

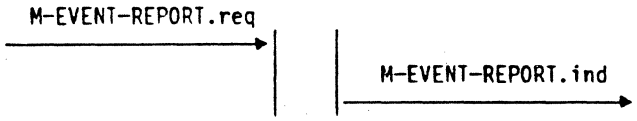


Figure F-8. Event Report Service Primitives

5. Confirmed Event Report - This is a service element used by a SMAE to report some unsolicited events of a resource to another SMAE, to which it expects a response. It has the following parameters:

   a. Invoke ID - optional, previously defined.

   b. Resource ID - required, previously defined.

   c. Event Type - This required field specifies the type of event being reported.

   d. Event Time - This required field contains the time of generation of the event.

   e. Event Info - This required field identifies information that the initiating SMAE is able to supply about the event.

   The response to Confirmed Event Report contains one of the following parameters:

a. Invoke ID - optional, previously defined.

b. Event Ack Info - This field identifies the success of the receipt of the Confirmed Event Report.

c. Status - If error is encountered during the operation, this required field contains the following error codes:

1) No Such Operation - previously defined

2) No Such Resource - previously defined

3) Access denied - previously defined

4) No Such Attribute - previously defined

5) Invalid Attribute Value - previously defined

Figure F-9 summarize the Confirmed Event Report service and its parameters:

parameter Name	Request	Response
Invoke ID	U	U
Resource ID	M	-
Event Type	M	-
Event Time	M	-
Event Info	M	-
Event Ack Info	-	M
Status	-	M

M - Mandatory
U - Un-mandatory, optional

Figure F-9. Confirmed Event Report Parameters

There are four sequence of service primitives associated with this service:

- M-CONFIRMED-EVENT-REPORT.req
- M-CONFIRMED-EVENT-REPORT.ind
- M-CONFIRMED-EVENT-REPORT.rsp
- M-CONFIRMED-EVENT-REPORT.conf

The sequence in which these primitives are used is illustrated below:

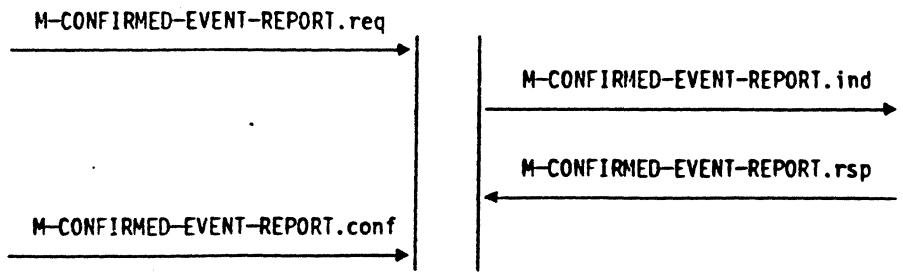


Figure F-10. Confirmed Event Report Service Primitives

Appendix G. CMIP/CMOL Comparison

The Intent of this section is to point out and detail where and how CMOL (CMIP over LLC) is described in this document, differs from CMIP (Common Management Information Protocol), as defined by ISO.

The information for this appendix was not available for this draft. It will be added before the next printing of this document.

Glossary of Terms and Abbreviations

If the definition was taken from an ISO document, the ISO document number is shown after the definitions (for example, 10040)..

List of Abbreviations

ACSE. Association Control Service Element

APPN. Advanced Program to Program Networking

AM. Attach Module

ASN.1. Abstract Syntax Notation 1

API. Application Program Interface

AVA. Attribute Value Assertion

CAU. controlled access unit

CCITT. International Telegraph and Telephone Consultative Committee

CMIP. Common Management Information Protocol

CMIS. Common Management Information Service

CMISE. Common Management Information Service Element

CMOL. CMIP over LLC

CM. Connection Manager

CNM. Communication Network Management

CPU. Central Processing Unit

CSMA/CD. Carrier Sense Multiple Access/Collision Detection. Ethernet media access method.

DIS. Draft International Standard

DIX. DEC**, INTEL**, XEROX**

DOS. Disk Operating System

DLC. Data Link Control

DU. Data User

EE. Entity Enabler

ED. Ending Delimiter

FC. Frame Control field

FCS. Frame Control Sequence

FS. Frame Status field

FSM. Finite State Machine

FDDI. Fiber Distributed Data Interface

FRMR. Frame Reject

HB. Heartbeat

HLM. Heterogeneous LAN Management

IAU. Intelligent Access Unit

ICI. Interface Control Information

IEEE. Institute of Electrical and Electronic Engineers

ID. Identifier

IP. Internet Protocol

ISO. International Standards Organization

LAN. Local Area Network

LANSM. LAN Station Manager

LF. Largest Frame

LLC. Logical Link Control

LME. Layer Management Element

LMN. LLC Name Management

LPDU. LLC Protocol Data Unit

LSA. Lower-Layer Services Architecture

LSAP. LLC Service Access Point

MAC. Medium Access Control

MEA. Managed Entity Application

MP. Managing Process

MIB. Managed Information Base

MPDU. Management Protocol Data Unit

MVS. Multiple Virtual Storage

NCP. Network Control Program

NEI. Network Entity Identifier

NEI Database. Collection of NEI names and other related information.

OS. Operating System

OSI. Open Systems Interconnect

PC. Personal Computer

PDU. Protocol Data Unit

RAS. Reliability, Availability, and Serviceability

RDN. Relative Distinguished Name

RI. Routing Information field

ROER. Remote Operations Error

RO. Remote Operations

ROIV. Remote Operations Invoke

RORS. Remote Operations Result

ROS. Remote Operations Services

ROSE. Remote Operations Service Element

RPU. Remote Program Update

SAA. Systems Application Architecture

SABME. Set Asynchronous Balance Mode Extended

SAP. Service Access Point

SD. Starting Delimiter

SDU. Service Data Unit

SLID. System Log Identifier

SMAE. System Management Application Entity

SMI. Structured Management Information

SNA. Systems Network Architecture

SNAP. Sub-Network Access Protocol

TCP. Terminal Control Program

UI. Unnumbered Information

UTC. Universal Time, Co-ordinated

VM. Virtual Machine

VTAM. Virtual Telecommunications Access Method

XID. Exchange Identification

Glossary

A

access unit id. Identifier for the CAU (controlled access unit).

attachment module. Module that connects the adapter to the access unit.

agent process. a management process capable of performing operations on managed objects and of issuing notifications on behalf of managed objects. (10040)

Application layer. management, directory, ACSE

application-entity (AE). The aspects of an application-process pertinent to OSI. (7498)

application-management. Functions in the application layer related to the management of OSI application-processes. (7498)

application-management-application-entry. An application-entity which executes application-management functions. (7498)

application-process. An element within a real open system which performs the information processing for a particular application.

Application processes can represent manual processes, computerized processes or physical processes. (7498)

application-service-element (ASE). That part of an application-entity which provides an OSI environment capability, using underlying services when appropriate. (7498-4)

Association Control Service Element (ACSE). ACSE provides a common framework for application communication by defining a standard method for all application protocols to open, release, and abnormally terminate application layer associations

attribute. properties of managed object that are visible at the object boundary. Each attribute has an associated value, which may have a simple or a complex structure; it may reflect or determine the behaviors or the managed object. The value of a simple attribute is observed or modified by an operation upon the object to read (GET), write (SET), or reset (DERIVE) the value, and additional operations (ADD and REMOVE) are defined for set- valued attributes.

attribute identifier. An identifier used to distinguish an attribute of a managed object class from all other attributes defined for that object class (10165-1)

attribute type. Defines a collection of values which an instance of that type may have, and a collection of operations (in their mathematical sense) which may be performed on values of that attribute type (10165-1)

Attribute Value Assertion (AVA). (10165-1),

B

Burned In Address. Default MAC address shipped with the device.

C

Common Management Information Protocol (CMIP). (9696-2)

Common Management Information Service Element (CMISE). (9695-2)

containment. part-of relationship. Used for naming

containment tree. a hierarchical arrangement of managed objects where the hierarchy is organized on the basis of the containment relationships. A managed object containing another managed object is higher in the hierarchy than the contained object, the containing managed object is referred to as being the SUPERIOR of the contained object, which is referred to as the SUBORDINATE. (10165-1)

D

distinguished name. (9594-2)

F

FRMR. Frame Reject is sent whenever an invalid command or response is received by a link station. station during which I-frame reception is suspended.

Functional Address. A subset of group addresses that allow multiple groups to be designated by a single address

G

Group Address. Address assigned to a collection of SAPs.

Group Identifier. A network entity identifier intentionally referring to potentially more than one instance. Multiple nodes of a particular network may be iden-

tically layered. So it makes sense to assign the same identifier to the same subset of functions in multiple nodes, although this need not be done. Thus, a group identifier may refer to multiple network entity instances. Examples of function subsets which can have group identifiers include APPN Network Node, OSI Intermediate System, PrintServer, etc. A network entity may have multiple group identifiers. A network entity may have both group identifiers and regular identifiers.

Group Find. A Find which is sent out to a group MAC address or a functional address. For example, a Find sent to the non-server functional address is a group Find.

H

HLM. Heterogeneous LAN Management

inheritance hierarchy. a hierarchical arrangement of managed object classes where the hierarchy is organized on the basis of the class refinement. A managed object class which is derived from another managed object class is lower in the hierarchy than the class from which it is derived. (10165-1)

I

I-format LPDU. A connection oriented physical data unit.

IAU (Intelligent Access Unit). An intelligent Token-Ring concentrator intended to provide enhanced levels of control and reliability for attaching workstations to IEEE 802.5 Token-Ring Local Area Networks. It consists of an intelligent access control unit and up to four lobe attachment modules.

L

layer-management. Functions related to the management of the (N)-layer partly performed in the (N)-layer itself according to the (N)-protocol of the layer (activities such as activation and error control) and partly performed as a subset of systems-management. (7498)

Link station. A protocol machine in an SNA node that manages the elements of procedure required for the exchange of data traffic with a communicating link station in an adjacent node.

Link station Id. The unique identifier for a Link Station.

lobe Id. Identification number of the cable that connects the adapter to the access unit.

local busy. A state that may occur for a given link

Local System Environment. The resources which exist in a real open system, but which are outside the OSI Environment. (7498-4)

LSAP Pair. A logical link connection (commonly known as a link station)

M

MAC address. The address of the device on the T-R

managed object. (1) The OSI Management view of a resource within the OSI Environment that may be managed through the use of OSI Management protocol(s). (7498-4) (2) Each part of the system that needs to be treated by management as an independent entity is represented by an instance of a managed object. (10164-1) (3) A managed object is defined by

- the properties or characteristics, termed ATTRIBUTES, visible at its boundary
- the MANAGEMENT OPERATIONS that may be applied to it
- the behavior exhibited by it in response to management operations, and
- the NOTIFICATIONS emitted by it.

managed object class. a named set of managed objects sharing the same set of attributes, notifications, management operations, and behavior (10040)

managed open system. a real open system supporting systems management (10040)

management application protocol data unit. an application protocol data unit exchanged between open systems for the purpose of systems management (10040)

management domain. a set of real open systems, collected either for functional or administrative purposes (10040)

management information. Information associated with a managed object that is operated on by OSI Management protocol to control and monitor that object. (7498-4)

Management Information Base (MIB). A conceptual composite of management information within an open system. That information within an open system which may be transferred or affected through the use of OSI Management protocols. The MIB encompasses information relating to managed objects. (7498-4)

management operation. a single act on a managed object to effect systems management. (10040)

management process. an application process participating in systems management (10040)

managing process. a management process capable of issuing operations and of receiving notifications (10040)

multicast. Broadcast message to selected group of workstations.

N

(N)-layer managed object. a managed object specific to the (N)-layer. (10040)

(N)-layer operation. The monitoring and control pertaining to a single instance of communication. (7498-4)

NEI Database. This database, which is contained in LANSM, is a collection of NEI names and other related information. LSA does not define this data base.

Network Entity. Defined in the *OSI Basic Reference Model*, ISO 7498-1984(E) as an active element within the network layer of the OSI layering. The Discovery architecture extends this definition of network entity to include an active element in any layer which sits on the DLC layer. A network entity can be thought of as any entity directly using the DLC. One network entity may communicate through multiple stations.

Network Entity Identifier. A permanent identifier for an entity. Thus, a network entity identifier identifies a subset of functions which interface with the DLC layer.

notification. (1) the act of informing about an event occurred in a managed object (10040) (2) Sets of information emitted by objects when certain events occur. What these notifications contain and the events that trigger them must be defined in the object definitions

O

object. (10165-1)

open system. The representation within the Reference Model of those aspects of a real open system that are pertinent to OSI. (7498)

OSI Environment. The resources which enable information processing systems to communicate openly, i.e., to conform to the services and protocols of open systems interconnection. (7498-4)

OSI management. The facilities to control, coordinate and monitor the resources which allow communications to take place in the OSI Environment. (7498-4)

OSI resources. Data processing and data communication resources which are of concern to OSI. (7498)

OSI/CS. IBM's one and only SAA system for providing OSI layers 4,5,6 protocols for the MVS and VM environments. It uses standard interfaces to strategic IBM systems software: it executes as a VTAM subsystem, uses NCP/NPSI or 9370 Telecommunications Subsystem for X.25 and communicates with NetView.

P

peer entities. Entities within the same layer (7498)

polymorphism. the ability of a managed object to be accessed as a member of any of several object classes (10165-1)

Presentation layer. Kernel, ASN.1

Primary Name. A unique name for the system in which a LAN Station Manager resides, which is used to name the LAN Station Manager. In cases where a universally-administered address (or burned-in address) exist, the primary name should be the universally-administered address.

upstream neighbor. For any station on a ring, the station that is sending frames or tokens directly to it.

product instance id. a subvector of the following MAC frames: Request Initialization, Report Ring Station Attachments, Report New Active Monitor. The field is 18 bytes long and is used to uniquely identify the hardware product instance of the attached product.

R

real open system. A real system which complies with the requirements of OSI standards in its communication with other real systems. (7498)

real system. A set of one or more computers, the associated software, peripherals, terminals, human operators, physical processes, information transfer means, etc., that forms an autonomous whole capable of performing information processing and/or information transfer. (7498)

Relative Distinguished Name (RDN). (9594-2)

Regular Identifier. A network entity identifier intended to refer to a single instance of the entity. Each regular identifier refers to one and only one instance of a network entity. Each network entity added to a Discovery instance must have at least one regular identifier.

ring access priority. - The maximum priority that a token can have for the adapter to use if for transmission

S

SAP. The logical point at which an n + 1 layer entity acquires the services of the n-layer.

segment number. number assigned to the ring for purposes of routing

Server. An entity which is identified by a group identifier. Such entities are Heartbeated.

specific management functional area. a category of systems management user requirements (10040)

Station. Consists of one DLC.LAN.MGR and the (possibly multiple) instances of the LLC and MAC sublayers and the Physical layers it manages. All these Physical layer instances must be attached to the same bridged LAN under non-failure conditions.

subclass. a class derived from another class by refinement (10165-1)

subordinate object. (10165-1)

superclass. a class used in deriving another class by refinement (10165-1)

superior object. See containment tree (10165-1)

systems management application services element. an application service element providing systems management services (10040)

systems-management. Functions in the Application Layer related to the management of various OSI resources and their status across all layers of the OSI architecture. (7498)

systems-management-application-entity (SMAE). An application entity for the purpose of systems management communication. (7498-4)

T

T1 timer. Reply Timer

T2 timer. Receiver Acknowledgment Timer

TI timer. Inactivity timer

U

Universal Group Identifier. A group identifier beginning with X'00'. Universal group identifiers are architected. A network entity may have only one universal group identifier.

W

Wall Plug. The wall connection to the access unit

X

XID. Exchange Identification

Index

D

E

F

S

T

Reader's Comments

IBM/3Com Heterogeneous LAN Management
Architecture Reference
Version 0.1.0

Publication No. SC30-3539-0

****

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

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

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

Phone No.