6.4.2 Attribute Template and Definition

Let us define an attribute template for the managed object class tokenRing.

```
tokenRingBandwidth ::= INTEGER
tokenRingCardPrice ::= SET OF INTEGER
tokenRingID ::= tokenRingAddress
tokenRingAddress ::= OCTET STRING SIZE (4)
tokenRingBandwidth              ATTRIBUTE
      WITH ATTRIBUTE SYNTAX INTEGER;
REGISTERED AS                   {1 3 5 8 9 2};
tokenRingCardPrice              ATTRIBUTE
      WITH ATTRIBUTE SYNTAX SET OF INTEGER;
REGISTERED AS                   {1 3 5 8 9 3};
tokenRingID                     ATTRIBUTE
      WITH ATTRIBUTE SYNTAX tokenRingAddress;
REGISTERED AS                   {1 3 5 8 9 4};
```

In the preceding definition of attributes for the managed object class tokenRing, observe the following interesting points:

- For defining the attributes, ASN.1 notations are used.
- tokenRingBandwidth is the template label, and ATTRIBUTE is the template name. The ATTRIBUTE template starts with tokenRingBandwidth and ends with ";" for REGISTERED AS.
- REGISTERED AS {1 3 5 8 9 2} is a *construct;* REGISTERED AS is the *construct name* and {1 3 5 8 9 2} is the *construct argument.*
- Formats for defining attributes follow a definite pattern, starting with the data type. This is followed by the definition of each attribute. Each attribute name has the keyword ATTRIBUTE. Then the data type of the attribute is defined by WITH ATTRIBUTE SYNTAX. The definition of the attribute ends with REGISTERED AS. These values are obtained either by internally defining the attributes or by following the guidelines of standards bodies. However, the values must be unique and are indicated by the last numbers within the braces: 2, 3, or 4. Note that the registration numbers used are arbitrary; these are used just for explanation purposes.
- The attribute identifier values given by REGISTERED AS are used to identify the attribute, and they are in addition to the managed object class identifiers.
- Notice that tokenRingID is defined in terms of another data type, tokenRingAddress. We again define the data type of tokenRing-

Address. Thus, the data type of tokenRingID is defined by the data type of tokenRingAddress.

An attribute template has some additional keywords, which will be explained next. Examine the attribute counter1.

```
counter1                ATTRIBUTE
     WITH ATTRIBUTE SYNTAX INTEGER;
     MATCHES FOR EQUALITY, ORDERING;;
REGISTERED AS           {1 3 5 8 9 5};
```

MATCHES FOR defines tests that can be performed on values of attributes for filter operation. If this is not indicated, then the tests on the values cannot be done, and they are not defined. Any specific attribute characteristics or indications of how attributes behave under matching rules are provided by MATCHES FOR or other behaviors. Behaviors that are specific to a managed object class are defined in the managed object class definition template.

The attribute counter1 is defined as an INTEGER that can take only a single value. MATCHES FOR is then defined; it means that we can test this counter for equality, that is, whether the counter value has reached or is equal to 30. By having the qualifier ORDERING, we can also test whether the present counter value is greater than or less than 30. There are two semicolons because there are two qualifiers, EQUALITY and ORDERING.

```
tokenRingCounter        ATTRIBUTE
     DERIVED FROM counter1;
REGISTERED AS           {1 3 5 8 9 6};
```

In the preceding definition of the attribute tokenRingCounter, attribute characteristics are derived from another attribute, counter1. Here, counter1 is defined in a different place. Note that DERIVED FROM is absent if the WITH ATTRIBUTE SYNTAX is present. DERIVED FROM helps us to make use of attribute definitions already made. By defining new rules, we can further extend or restrict the definitions derived from another attribute.

Let us proceed in our definition of counter by adding a new keyword, BEHAVIOR.

```
counter2            ATTRIBUTE
       WITH ATTRIBUTE SYNTAX INTEGER;
       MATCHES FOR EQUALITY, ORDERING;;
       BEHAVIOR
          counterBehavior  BEHAVIOR
          DEFINED AS "Tests for equality and greater than values
          are permitted.";
REGISTERED AS  {1 3 5 8 9 7};
```

DEFINED AS is used along with BEHAVIOR. DEFINED AS is followed by a string, and it supplies further meaning for the behavior of the managed object class, name bindings, parameters, attributes, actions, or notifications. Here, note that we have used quotes (" ") as text delimiters.

An attribute template uses the keyword PARAMETER. When an extension is required for the syntax of attributes, operations or notifications, PARAMETER is used to provide extensions. To define PARAMETER, we reference the label defined in a PARAMETER template (see Section 6.4.7).

```
counter3              ATTRIBUTE
       WITH ATTRIBUTE SYNTAX INTEGER;
       MATCHES FOR EQUALITY, ORDERING;;
       BEHAVIOR
          counterBehavior         BEHAVIOR
          DEFINED AS "Tests for equality and greater than values
          are permitted.";
       PARAMETER counterThresholdDetails;
REGISTERED AS                     {1 3 5 8 9 8};
```

In the preceding definition of the counter3 attribute template, we have added counterThresholdDetails along with the keyword PARAMETER. PARAMETER counterThresholdDetails is defined outside the counter3 attribute, and it is defined in the parameter template.

An attribute can have a single value. For TokenRingBandwidth, it may be 4. It can also have values of 16 and 100. Here we need to use the data type SET OF. Types can also be SEQUENCE and SEQUENCE OF.

The attribute values can be a set of values of the same data type. These are restricted to a *permitted set*. The permitted value set mentions the values that an attribute can take. They may also be called *allowed values*. When being modified, the value of an attribute cannot cross the limits of a permitted value set. A permitted value set can be further restricted to a *required value* set. This set can be empty if no values are required; otherwise, it mentions the values an attribute is required to have.

When a managed object class has many attributes, it may be better to subdivide it into subordinate classes. This will improve the efficiency of operations that are performed on it.

6.4.3 Attribute Group

A managed object class may have many attributes. For convenience and ease of operations, we can combine attributes into an *attribute group*. However, this restricts the operations we can perform on an attribute group. An attribute group has no value; hence, only those operations that do not require values can be performed on it.

Attribute groups are of two types: *fixed,* in which a collection of attributes is defined and more attributes cannot be added; and *extensible,* in which attributes can be added. Extensible attributes are defined in mandatory or conditional packages.

Attributes can be part of different attribute groups. In the following example, the attribute tokenRingID can be part of another attribute group in addition to the tokenRingGroup.

```
tokenRingGroup              ATTRIBUTE GROUP
GROUP ELEMENTS              tokenRingBandwidth, tokenRingCardPrice,
                            tokenRingID;
FIXED;
DESCRIPTION     "This includes tokenRingID of lanNet managed object
                class.";
REGISTERED AS               {1 3 5 8 9 6};
```

In the preceding definition of the attribute group tokenRingGroup, the attributes tokenRingBandwidth, tokenRingCardPrice, and tokenRingID are grouped together. By adding the keyword FIXED, we have ensured that no more attributes can be added to the tokenRingGroup. If FIXED was not there, we could add one or more attributes, such as tokenRingBridge. Note that individual attributes such as tokenRingBandwidth, tokenRingCardPrice, and tokenRingID can be single-valued or set-valued attributes.

6.4.4 Action Template

The *action template* is used in the definition of a managed object class. It maps to the action type parameter of CMIS M-ACTION service. For example, in a token ring, a ring station may have errors. When errors are hard errors or errors that must be rectified before a ring station becomes operational, there is no option but to bypass the ring station. When the errors are rectified, the ring station can be brought back into the token ring network. To bypass the ring station, a new action, tokenRingBypass, is defined as follows:

```
tokenRingBypass                 ACTION
      BEHAVIOR                  ringStationBypass;
      MODE CONFIRMED;
      WITH INFORMATION SYNTAX CHARACTER STRING SIZE (128);
      WITH REPLY SYNTAX         CHARACTER STRING SIZE (128);
REGISTERED AS                   {1 3 5 8 9 6};
ringStationBypass               BEHAVIOR
      DEFINED AS "When a ring station is to be bypassed on hard
                  errors, this message is sent.";
```

The preceding definition of tokenRingBypass furnishes some interesting details. tokenRingBypass is the name of the action defined. For details of the behavior of this action, we must go to the behavior template labeled "ringStationBypass." Because it contains the keyword MODE CONFIRMED, Action will have a confirmed message or reply after sending ACTION. If MODE CONFIRMED is not there, then ACTION is confirmed or unconfirmed as decided by the managing station.

WITH INFORMATION SYNTAX furnishes details of the information carried by Action. It is defined here as a string of messages which can be up to 128 octets. However, if this keyword is absent, then there is no information carried by Action.

The keyword WITH REPLY SYNTAX carries details of reply information that is sent as a result of Action. WITH REPLY SYNTAX is also defined as a string of messages with a maximum length of 128 octets. If this keyword is absent, then there is no reply associated with Action. Finally, REGISTERED AS is our usual identifier of the Action template.

6.4.5 Behavior Template

When a managed object is defined, it must be specified how attributes, operations on attributes, notifications, and name bindings behave. These details are explained in a behavior template. Behavior should be an extension of the earlier aspects of behaviors, to which it should not add new semantics or meanings.

```
tokenRingBridgeError          BEHAVIOR
      DEFINED AS "When a token ring bridge encounters errors, this
                  message is sent.";
```

In the definition of managed object classes, behavior is a text description that is prone to ambiguous interpretations and is not parsable by a machine. To overcome this limitation, formal languages such as Z, object Z, and specification and description language (SDL) can be used. As some of the languages are complex and their use involves placing an additional burden on modelers, the methodology of describing behavior using formal languages is not unanimously agreed upon (Reference 6.14).

6.4.6 Notification Template

A managed object sends notifications when a certain internal or external event occurs. The notification must contain information to be useful.

The kind of notification and the information it contains are defined when defining a managed object class. The notification type defined in the notification template is carried in the Event Information or Event Reply parameters of CMIS M-EVENT-REPORT.

For example, when a token ring adapter is about to fail, it may provide an indication or emit a message saying that it is going to be "dead." This can be defined as follows:

```
tokenRingBeaconing              NOTIFICATION
REGISTERED AS                   {1 3 5 8 9 7};
```

Whether these notifications are logged internally or forwarded externally depends upon the *event forwarding discriminators* (EFDs). EFDs also determine whether these notifications generate confirmed or unconfirmed event reports. Refer to Table 6-1 for more explanation of EFD.

6.4.7 Parameter Template

Parameters can be associated with attributes, operations, and notifications. They can be included in package, attribute, action, and notification templates. A parameter can define CMIS processing failures, notification requests and responses, or action requests and responses.

Referring to the example of counterThresholdDetails, a parameter template is defined as follows:

```
counterThresholdDetails         PARAMETER
       CONTEXT                  ACTION-REPLY;
       WITH SYNTAX              CHARACTER STRING (40);
REGISTERED AS                   {1 3 5 8 9 14};
```

After a threshold value of 30 is reached for soft errors in the network, the error message SoftErrors Threshold is exceeded by *x* number of errors, where *x* is any number from 0 to 29. However, when we reach the number 30 again, our counter wraps around. Soft errors are errors in ring stations in a token ring network where the ring station need not be bypassed.

In the preceding example, CONTEXT references conditions defined externally to this parameter template. As an example, CMIS parameters are defined in another place. The error message is carried in the CMIS M-ACTION Action Reply parameter. This parameter will be a Character String with a maximum of 40 octets.

6.4.8 Package Template

The *package template* is used for grouping many characteristics of a managed object class. If we look from the top of a managed object class template down, the package template is one hierarchy below the managed object class template. In managed object class templates, package templates can be included using CHARACTERIZED BY or CONDITIONAL PACKAGES.

A managed object class consists of package templates. These package templates can be *mandatory* or *conditional*. Mandatory packages (as the name suggests) are required. However, the attributes in conditional packages will be present depending on the conditions spelled out in the definitions. For example, if a printer is the managed object class, the toner package will be present if it is a laser printer.

These packages, in turn, consist of attributes visible at our conceptual boundary, operations on a managed object, behavior of a managed object, and the notifications emitted by a managed object class.

In packages, we have to consider the following key points:

- The rules regarding the creation and deletion of managed objects must be spelled out. The way managed object class instances relate to each other must be specified. If there is any relationship with other managed object class instances, these must be specified, too. Initial value managed object (IVMO) values are also outlined.
- The attributes and operations that can be done on these attributes should be indicated.
- Attributes have a property list that defines operations that can be done on them. The list can also supply details such as default values, initial values, permitted values, and required values. For example, the attribute tokenRingBandwidth can have a default value of 4 Mbits, if the network has 4 Mbits as the most common value. However, the permitted value can be anywhere from 0 to 100. The required values can be in the range of 1 to 16 Mbits. When there is a default value of 4 Mbits, we can set the initial values to 1 Mbit in another token ring that has mostly adapters with values of 1 Mbit.
- The operations and notifications of managed object class instances are specified. As noted earlier, operations on attributes are Get, Replace Attribute Values, Replace with Default Value, Add Member, and Remove Member. However, the operations on a managed object

class, such as Create and Delete, are part of the name binding template. Actions are defined in action templates.

Now, let us define a package by combining the concepts we introduced earlier.

```
tokenRingLan            PACKAGE
    BEHAVIOR            tokenRingLanBeh
    ATTRIBUTES          tokenRingBandWidth    REPLACE-WITH-DEFAULT,
                        tokenRingCardPrice    GET,
                        tokenRingID           PERMITTED VALUES
                                              000000-XXXXXXX,
                        tokenRingCounter      INITIAL      VALUE 0;
    ATTRIBUTE GROUPS    tokenRingGroup;
    ACTION              tokenRingBypass;
    NOTIFICATION        tokenRingBeaconing;
tokenRingLanBeh         BEHAVIOR
    DEFINED AS          "This LAN segment connects token ring";
```

In the preceding template, we have left out REGISTERED AS. If CONDITIONAL PACKAGES is to be used, then REGISTERED AS is required.

6.4.9 Name Binding Template

Name binding template is useful for defining the life cycles of managed objects. It defines the rules for creating, deleting, copying, and naming managed objects. Principles involved in naming are explained in X.720. Each instance of a managed object class needs to have a unique name. This is formed by concatenating the relative distinguished names (RDNs) indicated in the naming tree. An RDN is unique with respect to its superiors. The RDN consists of the naming attribute used in the name binding template and the value associated with it.

Return to the example of tokenRing. In the naming tree, lanNetwork (Chapter 4, Figure 4-9) is the superior managed object class, if we consider the tokenRing managed object class. To make up the name of a managed object instance, we concatenate RDNs. It is necessary for one attribute, known as a *distinguishing attribute*, to be unique in the managed object class tokenRing, and this is used to distinguish each instance.

Previously, we assumed that workstationIDs are uniquely assigned IDs for identification purposes in the tokenRing network. This enables the naming relationship of instances formed by using either networkAddress or workstationID.

```
tokenRingNaming                    NAME BINDING
     SUBORDINATE OBJECT CLASS      tokenRing AND SUBCLASSES;
     SUPERIOR OBJECT CLASS         lanNetwork;
     WITH ATTRIBUTE                workstationID;
     BEHAVIOR                      workstationIDnaming;
     CREATE                        WITH-AUTOMATIC-INSTANCE-NAMING;
     DELETE                        ONLY-IF-NO-CONTAINED-OBJECTS;
     REGISTERED AS                 {1 3 5 8 9 20};
workstationIDnaming                BEHAVIOR
     DEFINED AS                    "This is unique identifier";
```

In the preceding name binding template, there are many interesting issues. The distinguishing attribute is workstationID. It is used for forming the RDN of lanNetwork. However, for forming a unique name or a distinguished name, name binding of the superior managed object class lanNetwork is used. SUBORDINATE OBJECT CLASS is actually the label of the managed object class template. AND SUBCLASSES here states that workstationID can be used for naming of subclasses of the managed object class tokenRing. BEHAVIOR should specify the choices that must be made if there is more than one name binding relationship.

When creating an object instance, one need not specify a name for the Create object operation. In Delete, by specifying ONLY-IF-NO-CONTAINED OBJECTS, the Delete operations are limited or restricted. A managed object that is being deleted may have zero or more objects contained in it, so all the contained objects must be deleted before a Delete operation is performed; otherwise, there will be an error.

With another option, DELETES-CONTAINED-OBJECTS, it doesn't matter whether there are contained objects. They are deleted, too.

6.4.10 Managed Object Class Template

The definition of a managed object class is uniformly done in a standard template to avoid the confusion that might result from different people defining objects in different manners. This ensures that a managed object class defined in place A can be interpreted easily in place B. The good news is that we make use of templates we have previously defined. The managed object class template is at the top of the definitions hierarchy.

One of the important keywords is DERIVED FROM, which indicates that all the characteristics of the superclasses in the inheritance hierarchy will be inherited. The highest superclass is *top*. In other words, all managed objects are derived from the managed object class *top*. Also, all characteristics of superclasses are inherited, and none can be excluded. We can add more characteristics by including mandatory and conditional packages.

However, while defining managed object classes, one can develop one's own inheritance hierarchy instead of specializing from *top*. Notice that this may lead to the problem of interoperability with other systems.

The presence of mandatory packages is indicated by the keyword CHARACTERIZED BY. As seen earlier, packages may have behavior, attributes, attribute groups, operations, and notifications. The presence of conditional packages is identified by the keyword CONDITIONAL PACKAGES.

Making use of earlier definitions, a managed object class, tokenRing, is defined.

```
tokenRing                    MANAGED OBJECT CLASS
     DERIVED FROM            lanNet;
     CHARACTERIZED BY        tokenRingLan;
     CONDITIONAL PACKAGE     tokenRingRouter PRESENT IF "connected
                             to an FDDI backbone LAN";
     REGISTERED AS           {1 3 5 8 9 2};
```

In the preceding definition of the tokenRing managed object class, the superclass is lanNet, which, in turn, may have its own superclasses. The mandatory package is tokenRingLan, which has already been explained. There is, however, one more conditional package of tokenRingRouter. It specifies that this conditional package will be present only if the token ring network is connected to an FDDI backbone. In this case, definitions of tokenRingRouter will also be included. As mentioned earlier, REGISTERED AS gives the unique identifier for a managed object class.

Compilers are available for managed object class definitions. These compilers do the syntax checking on whether the managed objects defined follow the Guidelines for the Definition of Managed Objects (GDMO) format. These compilers furnish outputs in GDMO and ASN.1 formats. Some front-end editors take the user-friendly screen inputs for the definition of managed object classes in GDMO format. These editors do the syntax checking and provide definitions of managed object classes in GDMO format. This can be used as input to GDMO compilers.

We have already alluded to the possible ambiguities in the definition of managed object classes due to BEHAVIOR. Some more limitations/constraints of GDMO are:

- The informational model cannot be refined beyond a certain stage because of the object sizes and complexity of the object definitions.
- Interactions between objects are difficult to model. As a result, it is difficult to combine different views of an object.
- Modeling of same resources by different experts may result in different managed object class definitions. There are also cases where some prefer small managed object classes as compared to large complex managed object classes. Some ambiguities in design will occur, as the design of managed object classes is still an art.

6.5 Notes on the ITU-T SMI Documents

Important managed object classes are defined in X.721, Definition of Management Information. Refer to Table 6-1 for an explanation of the MOCs defined in X.721. This is a core group of MOCs, and they can be used for defining other managed object classes used in TMN. Note that for explanation of the syntax of some of the MOCs defined in X.721, we have to further refer to other ITU recommendations furnished in references 6.5, 6.6, 6.7, 6.8, and 6.9. As an example, while discriminator and event forwarding discriminator MOCs are defined in X.721, their syntax and event report processing models are explained in X.734, Event Report Management Function (Reference 6.7). X.721 also defines attribute types, name bindings, packages, attributes, action types, parameter types, and notification types. In addition, conventions for conformance and templates for compliance testing are specified in this document.

Packages defined in X.721 are presented in Table 6-4. This document has attribute types such as counter, gauge, counter-Threshold, gauge-Threshold, and tideMark. These are useful for performance management. Twenty-five EVENT-REPORT—related attributes are also defined in this document; these are useful for different systems management functions. In addition, state and relationship attributes are defined in this document. X.721 is quite useful for defining our own managed object classes. It is, however, better to reuse definitions furnished in the document.

X.723 defines managed object classes that can be used as superclasses for defining the managed object classes of individual layers. These managed object classes, which are listed in Table 6-2, are known as *generic managed object classes*. Name bindings to be used are furnished in this document. These managed object classes can use inheritance and extend the definitions of managed object classes defined in X.723. Some of the objectives is to reduce duplication of efforts and inconsistencies in the definition of managed object classes.

In addition, 28 attributes used in the definition of managed object classes have been defined in document X.723. One attribute group (Counters), three Actions (activate, deactivate, and deactivateWhenNoUsers), and one notification (communicationsInformation) have been defined. Note that clProtocolMachine can be used by an entity using a connectionless mode communications function. Similarly, coProtocolMachine refers to a connection-oriented communication function.

TABLE 6-1

Managed Object Classes Defined in X.721 and Their Descriptions.

Managed Object Class	Description
alarmRecord	Used to define the format of logs for storing alarm notifications or alarm reports. alarmRecord is a subclass of eventLogRecord.
attributeValueChangeRecord	Defines the format of logs for storing notifications or event reports due to attribute value change. The attributeValueChangeDefinition attribute is a SET OF identifier of the attribute whose value has changed, the old attribute value, and a new attribute value. Note that these values are read-only. The conditional package sourceIndicatorPackage indicates whether the attribute value change is due to an internal operation, a management operation, or an unknown source. Also, the conditional package attributeIdentifierListPackage provides the set of attribute Ids of attributes whose value are being changed. This MOC is a subclass of eventLogReport.
discriminator	A superclass MOC introduced to control event reporting to external systems. The attribute discriminatorConstruct furnishes the logical operations, which should be evaluated to TRUE, to forward event reports. discriminatorConstruct functions as a filter and determines which event reports are to be forwarded. Conditional scheduling packages are used to control the scheduling of the event reports.
eventForwardingDiscriminator	A subclass of discriminator. As this MOC is inherited from discriminator MOC, eventForwardingDiscriminator sets conditions that should be satisfied before event reports are sent to one or more destinations. If the active destination/ destinations fail, then a backup destination list is used to forward the event reports. The attribute confirmedMode indicates whether the receipt of event report has to be acknowledged or not.
eventLogRecord	A subclass of logRecord. This MOC defines the log records in a log due to event reports or notifications. The log records can be classified by event types.
log	Derived from top; used to control the logging of information. Log is a collection of records. The logging behavior is controlled by the discriminatorConstruct attribute. logFullAction attribute states whether the oldest log record is deleted (wrap) or to halt logging when the maximum log size is reached. Conditional scheduling packages control the duration and start time of logging. The logging may be done by an external system, if the externalScheduler conditional package is present. maxLogSize attribute specifies the size of the log in octets. current-LogSize attribute indicates the current size of a log.
logRecord	A subclass of top. logRecord managed objects are created for event reports or notifications and represent the information stored in logs. loggingTime indicates the time when a log record was stored in the log. logRecordId is a unique integer to identify a log record.
objectCreationRecord	A subclass of eventLogRecord. This MOC defines the format of logs for storing object creation notifications or event reports. The conditional package sourceIndicatorPackage indicates whether the managed object has been created due to an internal operation, a management operation, or an unknown source. Also the conditional package attributeIdentifierListPackage provides the list of attributes and values when a managed object is created.

TABLE 6-1 (Continued)

Managed Object Classes Defined in X.721 and Their Descriptions.

Managed Object Class	Description
objectDeletionRecord	Similar in functionality to objectCreationRecord, except that this notification or event report is received when a managed object is deleted instead of during creation of a managed object.
relationshipChangeRecord	Derived from eventLogRecord. This MOC is used for defining format of logs for relationship change notifications or relationship change event reports. This MOC also includes the conditional packages sourceIndicatorPackage and attributeIdentifierListPackage. We have discussed the functions of these packages in attributeValueChangeRecord. The attribute relationshipChangeDefinition has a similar syntax to that of attributeValueChangeDefinition.
securityAlarmReportRecord	Used for the specific purpose of logging when security alarm notifications or security alarm reports are received. The security alarm cause and severity, and the system that detects the alarm, are logged. Also, the alarm service requester and service provider are logged.
stateChangeRecord	Derived from eventLogRecord. This MOC is used for representing log records due to state changes such as object creation, object deletion, relationship change, or attribute value change log records. This MOC has also the conditional packages sourceIndicatorPackage and attributeIdentifierListPackage. The syntax of the attribute stateChangeDefinition is similar to that of the attribute ValueChangeDefinition.
system	Used to represent hardware or software used in information processing or information transfer. This MOC can be used for naming other managed objects. The mandatory attribute systemId, which can be used for uniquely identifying the system managed objects, can be a graphic string, integer, or NULL. Another mandatory attribute, systemTitle, which can also be used for identifying the system managed object, can be a distinguished name, object identifier, or NULL. The value of NULL is used when the system managed object is not configured or when the attribute is not to be used for naming.
top	The ultimate superclass in the class hierarchy of MOCs. Other MOCs involved in information processing or information transfer hardware or software are specialized from the top or MOCs specialized from top. top cannot be instantiated.

TABLE 6-2

Packages Defined in ITU-T X.721.

additionalInformationPackage	notificationIdentifierPackage
additionalTextPackage	dailyScheduling
attributeIdentifierListPackage	duration
attributeListPackage	externalScheduler
availabilityStatusPackage	sourceIndicatorPackage
correlatedNotificationsPackage	weeklyScheduling

6.6 TMN Information Model

Managed object classes for telecommunications resources such as telecommunications equipment and telecommunications services are defined in ITU-T Recommendation M.3100, Generic Network Information Model (Reference 6.10). M.3100 contains MOCs defined for physical resources and logical resources. Physical resources can be PBXs, digital cross-connect systems, cards, and shelves. These are usually represented as NEs. Logical resources can be communication protocols, logs, and network services. The MOCs in M.3100 are defined using the GDMO. The MOCs, which are similar, are grouped into different groups for convenience; these groups are known as *fragments*. The MOCs defined in M.3100 are explained in Table 6-3.

TABLE 6-3

Generic Managed Object Class Definitions in X.723.

applicationProcess	port
communicationsEntity	sap1
communicationsInformationRecord	sap2
clProtocolMachine	singlePeerConnection
coProtocolMachine	subSystem
physicalMedia	

TABLE 6-4

M.3100 Managed Object Classes and Their Descriptions.

Managed Object Class	Description
	Network Fragment
network	Represents a collection of telecommunications and management managed objects associated with a single administrative entity. May be owned by a single customer or provider, or associated with a specific service network.
networkR1	A subclass of network MOC.
	Managed Element Fragment
circuitPack	Used to model a plug-in replaceable unit that can be inserted or removed from an equipment holder. Line cards, processors, or power supply units are examples of resources that can be modeled as circuitPacks. The attributes associated with the MOC, such as availabilityStatus, indicate whether the correct circuit pack is installed. Similarly, administrativeState attribute is used to control the operation of a circuitPack, and operationalState attribute permits us to know whether an instance of the circuitPack is working or not. In addition to the attributes, there are notifications for changes such as creation or deletion of circuitPack instances. These are some of the basic principles of MOC manipulation.
equipment	Represents physical components including replaceable units. An equipment MOC may be nested within another equipment MOC.
equipmentHolder	Used to represent resources that can hold physical resources.
equipmentR1	A subclass of equipment MOC.
managedElement	Represents a network element. A managed element communicates with a manager/OS using Q interfaces.
managedElementR1	A subclass of managed element.
managedElementComplex	Represents a collection of NEs. This grouping facilitates easy management of NEs by an OS.
software	Used to represent logical information stored in equipment. Can represent software or firmware used in equipment and data tables.
softwareR1	A subclass of software.
	Termination Point Fragment
connectionTermination-PointBidirectional	Managed object originates and terminates a link connection.
connectionTermination-PointSink	Managed object terminates a link connection.
connectionTermination-PointSource	Managed object originates a link connection.
terminationPoint	Represents termination of a transport entity such as a trail or connection. One or more connections, when linked together, form a trail. A trail includes two trail termination points, one or more connections, and associated connection termination points. A trail can be either unidirectional or bidirectional.

TABLE 6-4 (Continued)

M.3100 Managed Object Classes and Their Descriptions.

Managed Object Class	Description
trailTerminationPoint-Bidirectional	Used to represent a termination point where one trail terminates and another trail originates.
trailTerminationPointSink	Represents a termination point where a trail terminates.
trailTerminationPointSource	Used to model a termination point where a trail originates.
	Switching and Transmission Fragment
circuitEndPointSubGroup	Represents a circuit end point that connects one exchange to another.
connectionR1	Represents a connection used to transparently transfer infromation between connection termination points. Connection is a component of a trail.
pipe	Represents managed objects responsible for transfer of information between termination points. The information transfer is between a trail and connection. Trail and connection have a client-server relationship.
trialR1	A subclass of pipe. Managed objects of this MOC are responsible for the integrity of information transfer in a trail.
	Cross-Connection Fragment
crossConnection	Used to represent a connection that connects two termination points. Different types of cross-connections are furnished in Appendix I of M.3100.
fabric	Manages establishing and releasing of cross-connections. Also, managed objects belonging to the fabric MOC are responsible for creating, removing, and modifying termination points available in termination point pools and group termination points (GTPs).
fabricR1	A subclass of fabric.
gtp	Represents a group of termination points managed as a single entity.
mpCrossConnection	Assigns relationship between termination points or group termination points. From termination is furnished by the fromTermination attribute and to termination is indicated by the toTermination attribute in the contained crossConnection managed objects.
namedCrossConnection	A subclass of crossConnection. Used for managing sensitive cross-connections. An example is the line to the President of the United States.
namedMpCrossConnection	A subclass of mpCrossConnection. This MOC is also used for managing sensitive multiple cross-connections.
tpPool	Represents a set of termination points or GTPs.
	Functional Area Fragment
alarmRecord	Defined in X.721.
alarmSeverity-AssignmentProfile	Specifies the alarm severity assigned to managed objects.

The MOCs defined in M.3100 are generic and support a broad behavior. MOCs required for specific telecommunications resources have to be specialized from the MOCs defined in M.3100. Many TMN application-specific managed object classes are defined in ITU-T recommendations such as Q821, Alarm Surveillance (Reference 6.11); Q822, Performance Management (Reference 6.12); and Q823, Traffic Management (Reference 6.13). In addition to these, standards bodies such as ANSI and ETSI have defined their own managed object classes for specific purposes. Table 6-5 furnishes summaries of the MOCs defined in Q821 and Table 6-6 provides explanations of MOCs defined in Q822.

As a rule of thumb, before defining new managed object classes, examine the availability of standard managed object classes. As an example, to define performance managed object classes, use standard managed object classes, such as scanner, defined in X.739, Metric Objects and Attributes. However, if the standard managed object classes do not satisfy your specific needs, then specialize the managed object class from the standard managed object classes.

Note that managed object classes use CMIP as the management protocol. FTAM can also be used in some cases.

TABLE 6-5

Description of Managed Object Classes in Q.821, Alarm Surveillance.

Managed Object Class	Description
currentAlarm-SummaryControl	Used for the generation of current alarm summary reports. This MOC is a subclass of top. The attribute objectList provides the list of managed objects for which current alarm summaries have to be generated. The type of current alarm summary can be controlled on the basis of the alarm status, perceived severity, and probable cause. The scheduling of the current alarm summary reports is done by the management-OperationsSchedule managed object class.
managementOperations-Schedule	Provides the functionality to schedule periodic management services on specified managed objects. The starting and ending times of the scheduling activity are furnished by the beginTime and endTime attributes, respectively. Attribute interval provides the time between the occurrences of scheduling activities.

TABLE 6-6

Description of Managed Object Classes in Q.822, Performance Management.

Managed Object Class	Description
currentData	Contains current performance data. This MOC is derived from scanner MOC defined in X.739. To create current performance data, the currentData managed object is assumed to be contained in a managed object such as circuitPack. Attributes monitored are in the form of counters or gauges. currentData has conditional packages to ensure creation of history data and to send scan report notifications. History data are formed from current performance data at the end of each performance reporting interval as furnished by granularityPeriod. If conditional package thresholdPkg is included, then quality of service alarm notifications are sent to managing systems when threshold limits are crossed.
historyData	Derived from top. At the end of each current performance data collection interval, a copy of the current data is stored in an instance of the historyData. Note that the creation of the historyData managed object is optional and a historyData managed object is created if discriminatorConstruct in the conditional package, filterSuppressionPkg, evaluates to TRUE. filterSuppressionPkg is a conditional package in currentData MOC.
thresholdData	Specialized from top. Contains values of counter or gauge threshold settings.

6.7 Example of GDMO and ASN.1 Definitions

Here we furnish an example of how GDMO and ASN.1 definitions for managed object classes are made. The primary motivation for this example is to define a managed object class showing the steps involved in the definitions of a MOC. The syntax checking can be performed with GDMO and ASN.1 compilers.

In this simplistic example, managedElementDiagnosticTest is the MOC designed to perform diagnostic test on an NE. This test is performed from an OS. Note that we have deliberately borrowed the ASN.1 definitions from X.721 and other standard documents. In real life, it is not necessary to include those definitions, which are defined in other modules. Instead, while performing ASN.1 compilations, the modules are imported.

Basically, managedElementDiagnosticTest includes only one package—managedElementDiagnosticTestPkg. This package includes the attribute diagnosticTestDestination, which indicates the NE on which the diagnostic test has to be performed.

This package also includes two ACTIONS—performDiagnosticTest and abortDiagnosticTest. ACTION performDiagnosticTest is used to initiate the diagnostic test on the destination indicated by the attribute diagnosticTestDestination. ACTION abortDiagnosticTest aborts a diagnostic test in progress. Usually, it is essential to have a mechanism to end an ACTION, which performs starting of some actions. These two ACTIONS perform the functions required, and the replies sent do not contain much information except whether ACTION command has been accepted or not and the reasons for accepting or rejecting the ACTIONs. The detailed results of ACTION performDiagnosticTest are sent in the notification diagnosticTestResult. Notice that values in REGISTERED AS are arbitrary.

Notice that in the example furnished below, module definitions for GDMO and ASN.1 are not furnished.

6.7.1 GDMO Definitions

```
managedElementDiagnosticTest MANAGED OBJECT CLASS
      DERIVED FROM Recommendation X.721 | ISO/IEC 10165-2: 1992"
      :top;
      CHARACTERIZED BY
            managedElementDiagnosticTestPkg;
      REGISTERED AS {2 9 3 4 3 0}

managedElementDiagnosticTestPkg PACKAGE
      BEHAVIOR managedElementDiagnosticTestBeh;
      ATTRIBUTES
            diagnosticTestDestination
                  GET-REPLACE;
      ACTIONS
            performDiagnosticTest,
            abortDiagnosticTest;
      NOTIFICATIONS
            diagnosticTestResult;
REGISTERED AS {2 9 3 4 4 1}

managedElementDiagnosticTestBeh      BEHAVIOR
      DEFINED AS
            "This package is defined to perform and abort
            diagnostic tests on a managed element or NE. This test
            is run from an OS."

diagnosticTestDestination ATTRIBUTE
      WITH ATTRIBUTE SYNTAX DiagnosticTestDestination;
MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
BEHAVIOR diagnosticTestDestinationBeh;
REGISTERED AS {2 9 3 4 7 1}

diagnosticTestDestinationBeh BEHAVIOR
      DEFINED AS
            "This attribute identifies a network element and has
            the syntax of NameType. NameType is defined in M.3100."
```

```
abortDiagnosticTest           ACTION
      BEHAVIOR                abortDiagnosticTestBeh;
      WITH INFORMATION SYNTAX AbortDiagnosticTestInfo;
      WITH REPLY SYNTAX       AbortDiagnosticTestReply;
REGISTERED AS {2 9 3 4 9 1}

abortDiagnosticTestBeh      BEHAVIOR
      DEFINED AS
            "This action is used for aborting a diagnostic test on
            a NE which has been started by performDaignosticTest
            ACTION."

performDiagnosticTest         ACTION
      BEHAVIOR                performDiagnosticTestBeh;
      WITH INFORMATION SYNTAX PerformDiagnosticTestInfo;
      WITH REPLY SYNTAX       PerformDiagnosticTestReply;
REGISTERED AS {2 9 3 4 9 0}

performDiagnosticTestBeh    BEHAVIOR
      DEFINED AS
            "This action is used for performing diagnostic tests on
            a NE."

diagnosticTestResult          NOTIFICATION
      BEHAVIOR                diagnosticTestResultBeh;
      WITH INFORMATION SYNTAX DiagnosticTestResultInfo;
REGISTERED AS {2 9 3 4 10 0}

diagnosticTestResultBeh     BEHAVIOR
      DEFINED AS
            "This notification is used to send the results of a
            diagnostic test on a NE."

managedElementDiagnosticTest-managedElement            NAME BINDING
      SUBORDINATE OBJECT CLASS     managedElementDiagnosticTest and
      SUBCLASSES NAMED BY SUPERIOR OBJECT CLASS     managedElement;
      WITH ATTRIBUTE                                managedElementId;
      BEHAVIOR      managedElementDiagnosticTest-managedElementBeh;
      CREATE;
DELETE ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS {2 9 3 4 6 0}

managedElementDiagnosticTest-managedElementBeh BEHAVIOR
      DEFINED AS
            "This is for name binding of the managed objects diag-
            nostic tests. Diagnostic test objects can be created
            and deleted if there are no contained objects. man-
            agedElementId is defined in M.3100."
```

6.7.2 ASN.1 Definitions

```
AbortDiagnosticTestInfo ::=     SEQUENCE {
      diagnosticTestDestination       [0] DiagnosticTestDestination,
      additionalInformation           [1] AdditionalInformation}
```

```
AbortDiagnosticTestReply ::=    SEQUENCE {
       acceptedOrRejectedReason      [0] AcceptedOrRejectedReason,
       additionalInformation         [1] AdditionalInformation}

AcceptedOrRejectedReason           ::= ENUMERATED {
                                   accepted (0),
                                   rejected (1),
                                   notAllowedInCurrentState (2),
                                   diagnosticTestInProgress (3),
                                   diagnosticTestIsNotBeingDone (4),
                                   diagnosticTestNotSupported (5)}

AdditionalInformation ::= SET OF ManagementExtension

DiagnosticTestDestination ::= NameType

DiagnosticTestResultInfo ::=  SEQUENCE {
      acceptedOrRejectedReason     [0] AcceptedOrRejectedReason,
      additionText                 [1] GraphicString,
      additionalInformation        [2] AdditionalInformation
                                       OPTIONAL
}

ManagementExtension ::= SEQUENCE {
                  identifier        OBJECT IDENTIFIER,
                  significance      [1] BOOLEAN DEFAULT FALSE,
                  information       [2] ANY DEFINED BY Identifier }

NameType ::= CHOICE { numericName    INTEGER,
                      pSting     GraphicString}

PerformDiagnosticTestInfo ::=  SEQUENCE {
       diagnosticTestDestination     [0] DiagnosticTestDestination,
       additionalInformation         [1] AdditionalInformation }

PerformDiagnosticTestReply ::= SEQUENCE {
       acceptedOrRejectReason        [0] AcceptedOrRejectReason,
       additionalInformation         [1] AdditionalInformation}
```


6.8 Summary

In this chapter, we have discussed how a managed object class is defined using templates. This information is furnished in ITU-T X.722, Guidelines for the Definition of Managed Objects (GDMO). Along with this, we have examined the different templates, such as attribute, action, attribute group, notification, and behavior. An overview of ITU-T X.721 and X.723 has also been presented. We have discussed M.3100, as it is an important part of the TMN information model. We have ended the chapter by providing an

example of defining a hypothetical managed object class to explain the processes and steps involved in defining a managed object class.

6.9 References

6.1. ITU-T Recommendation X.720 (ISO 10165-1), Information Technology, Open Systems Interconnection, Structure of Management Information, Part 1: Management Information Model, 1992.

6.2. ITU-T Recommendation X.721 (ISO 10165-2), Information Technology, Open Systems Interconnection, Structure of Management Information, Part 2: Definition of Management Information, 1992.

6.3. ITU-T Recommendation X.722 (ISO 10165-4), Information Technology, Open Systems Interconnection, Structure of Management Information, Part 4: Guidelines for the Definition of Managed Objects, 1992.

6.4. ITU-T Recommendation X.723 (ISO 10165-5), Information Technology, Open Systems Interconnection, Structure of Management Information, Part 5: Generic Management Information, 1993.

6.5. ITU-T Recommendation X.730 (ISO 10164-1), Information Technology, Open Systems Interconnection, Systems Management: Object Management Function, 1992.

6.6. ITU-T Recommendation X.733 (ISO 10164-4), Information Technology, Open Systems Interconnection, Systems Management: Alarm Reporting Function, 1992.

6.7. ITU-T Recommendation X.734 (ISO 10164-5), Information Technology, Open Systems Interconnection, Systems Management: Event Report Management Function, 1992.

6.8. ITU-T Recommendation X.735 (ISO 10164-6), Information Technology, Open Systems Interconnection, Systems Management: Log Control Function, 1992.

6.9. ITU-T Recommendation X.736 (ISO 10164-7), Information Technology, Open Systems Interconnection, Systems Management: Security Alarm Reporting Function, 1992.

6.10. ITU-T Recommendation M.3100, Generic Network Information Model, 1995.

6.11. ITU-T Recommendation Q821, Stage 2 and Stage 3 Description for the O3 Interface—Alarm Surveillance, 1993.

6.12. ITU-T Recommendation Q822, Stage 1, Stage 2, Stage 3 Description for the O3 Interface—Performance Management, 1994.

6.13. ITU-T Recommendation Q823, Stage 2 and Stage 3 Functional Specifications for Traffic Management, 1996.

6.14. Raman, L., Information Modeling and Its Role in Network Management. In *The Telecommunications Network Management, Technologies and Implementations,* Aidarous, S. and Plevyak, T. (eds.), New York: The Institute of Electrical and Electronics Engineers, Inc., pp. 1—62, 1998.

This page intentionally left blank.

CHAPTER 7

ACSE, ROSE, CMISE, and CMIP

7.1 Introduction

In Chapter 1, Section 1.8, we discussed important concepts such as managers and agents. Management information must be communicated between a manager and agents. Without this, a manager cannot know what is happening in the objects under the control of agents. To build any useful system management functional application, a manager needs to gather management information on objects and, sometimes, store it. For gathering this management information, we will examine how associations between managers and agents are made.

Also, there are management protocols between manager and agents that must be explored. How frames are formed and carried between a manager and agents is examined in this chapter. We must note here that ITU-T recommendations furnish generic descriptions of services and protocols and rules for communications between peer systems. It is up to users to interpret these services and protocols and package them into meaningful profiles of management protocols between manager and agents. ITU-T recommendations also furnish generic definitions, such as those for service providers and service users. However, these terms have to be tailored to suit the TMN and system management requirements.

7.2 Application Layer Component Concepts

The topmost layer of the OSI reference model is the *application layer*. The application layer functions and the functions below it are used by the *application process* (AP). Figure 7-1 illustrates the concept of an AP. An AP combines all the information processing and communication aspects that are grouped together and given a single name for remote reference. An example of an AP is the retrieval of information from a database. This database may be in another open system, so retrieval will include different aspects such as establishing an association and communicating the command, in turn consisting of querying, receiving the response, and processing and presenting the response.

One AP may use one or more *application entities* (AEs) to represent its communications aspects. The application layer appears as a collection of AEs, each of which includes information for communicating with another peer AE. AEs use the immediate lower layer of presentation services

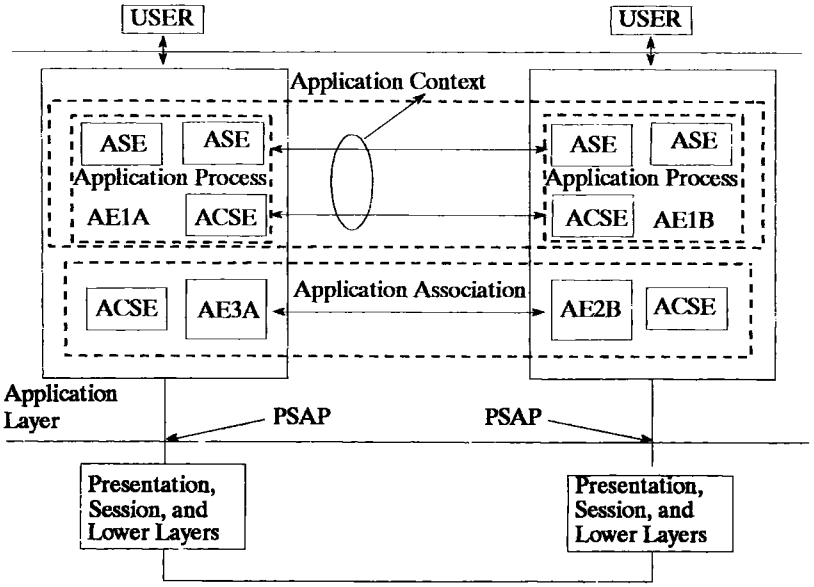


Figure 7-1
Concepts of application process, application entity, application context, and application association.

to communicate with each other. A cooperative logical connection between two AEs is known as an *application association* (AA). See Figure 7-1 for a depiction of an AA.

Functions of AEs required for cooperative processing are broken down further into *application service elements* (ASEs). These ASEs represent a collection of communication capabilities packaged into a module. One ASE may communicate with another ASE in the same application layer or with ASEs in other peer open systems using lower layers. These have their own service definitions and protocol specifications.

There are generic ASEs such as the *association control service element* (ACSE) and the *remote operations service element* (ROSE), as well as specialized ASEs such as the *common management information service element* (CMISE), the *systems management application service element* (SMASE), and *file transfer, access, and management* (FTAM). The *reliable transfer service element* (RTSE) is used for bulk data transfers. Because RTSE is not much used in TMN, we will not discuss it here. For details on RTSE refer to X.218, Reliable Transfer: Model and Service Definition (Reference 7.9) and X.228, Reliable Transfer: Protocol Specification (Reference 7.10).

There are advantages to classifying functions into generic ASEs. For example, if the establishment of an association for the transfer of messages such as in e-mail or a message handling system (MHS) is required, an ACSE can be used to establish the connection instead of "reinventing the wheel." Then, for any specific portion of the MHS, an MHS ASE can be used.

For a manager and agents to communicate, each must know about the other; thus the application context (AC) must be known. AC refers to the ASEs used between a manager and agents and the protocols used. For interoperability, ASEs and the rules for using them are grouped into identifiable ACs (see Figure 7-1). This application context is identified by an *application context name,* which is an object identifier. An application context assists in the cooperative working of AEs. This is one of the important parameters used during association or connection establishment. A run-time instance of an ASE is known as an *ASE invocation.* A collection of these invocations is known as a *control function* (CF).

7.3 Systems Management Service Elements

As we will refer to protocol data unit (PDUs) often, let us investigate what a PDU is. A PDU is a unit of data associated with a protocol. A PDU has protocol control information (PCI), which contains information to coordinate operation between two N entities, and possibly user data as shown in Figure 7-2. In the case of service data unit (SDU), the data are preserved during data transfer between peer (N + 1) entities and are not interpreted by the N entities.

When PDUs are associated with a particular protocol, they are given unique names. As an example, when PDUs are associated with application layer, these PDUs are known as application protocol data units (APDU).

ASEs such as ACSE, ROSE, CMISE, and SMASE are used by a systems management application entity (SMAE), as shown in Figure 7-3. SMASE is used to form *management application protocol data units* (MAPDUs). These

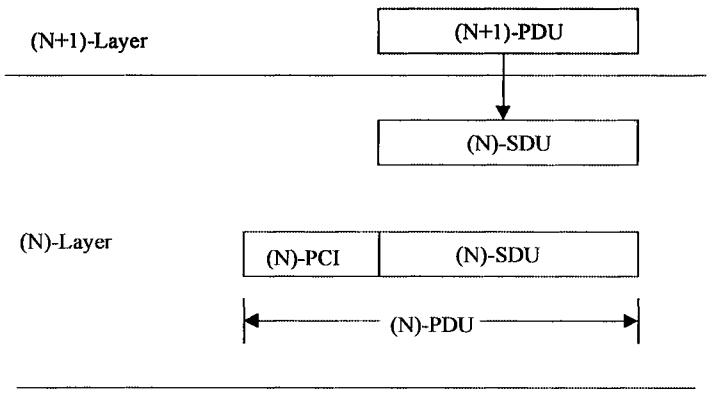


Figure 7-2
Concepts of a PDU.

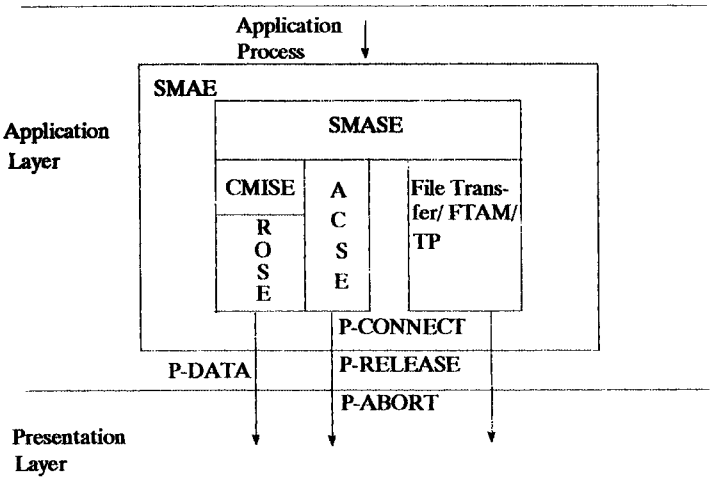


Figure 7-3
Relationship between systems management service elements.

MAPDUs as well as pass-through CMIP APDUs carry management information from one SMAE to another. The communication of management information between SMAEs is carried out by the *common management information protocol* (CMIP), or possibly by means of other communication services provided by ASEs such as *file transfer* or *transaction processing* (TP).

In TP, the key concept is *transaction*. The properties of a transaction are atomicity, consistency, isolation, and durability. *Atomicity* means that either all operations of a unit of work are performed or none of the operations are performed. *Consistency* refers to the units of work being performed accurately, correctly, and with validity. By *isolation*, we mean that partial results of a unit of work are not available. *Durability* represents the characteristic whereby a failure or any other action does not affect the results of the unit of work performed.

SMASE uses CMIS services. This frame is formed in CMISE, then transformed into the CMIP protocol frame in the *common management information protocol machine* (CMIPM). Here CMIPM is a *finite state machine* (FSM). The CMIP APDU has the appropriate ROSE headers to form the frame suitable for transfer of management data via underlying presentation services to another SMAE.

However, initially, for communication to take place between SMAEs, an association must be formed. This is a rather complex operation. The two SMAEs involved in forming an association must support similar *functional units*. A functional unit is an abstract concept used for combining service options. These functional units assist in providing systems management services. It may be noted here that there are functional units for the presentation layer as well as for the session layer. If functional units

between two SMAEs are similar, there is not much of a problem. But there may be cases where functional units supported by SMAEs are not similar. For this reason, during association, the characteristics of the association need to be negotiated. This is done with the help of ACSE services. Successful interoperability is improved by the use of a standardized application context and standard profile that specifies common application, presentation, and session layer options.

7.4 ACSE Services

For systems management purposes, management information needs to be transferred between a manager and agents. For data transfer between a manager and agents, a connection or association must be established initially. In addition, there should be the ability to release the connection when a manager and an agent do not want to communicate any longer. The ACSE services used for establishing an association and, subsequently, releasing the association formed are shown in Table 7-1. Release of an association means orderly closing and, in some cases, aborting the association. The prefix A- is added to ACSE services to distinguish them from other application layer services.

ACSE protocols are described in X.217, Service Definition for the Association Control Service Element (Reference 7.1). Authentication for connectionless ACSE was not available in X.217 service definitions; therefore, it was added in Amendment 1 to X.217 (Reference 7.2). Connection-oriented ACSE protocols are described in X.227 (Reference 7.3) and the connectionless ACSE protocols are explained in X.237 (Reference 7.5). In ASN.1 definitions some holes are left to provide for the possible extensions to the connection-oriented and connectionless ACSE protocols; these are provided by amendments to X.227 (Reference 7.4) and X.237 (Reference 7.6). Also, the amendment to X.237 (Reference 7.6) includes authentication parameters.

TABLE 7-1

Application Control Element (ACSE) Services.

Communication Mode	Service	Type
Connection-oriented	A-ASSOCIATE	Confirmed
	A-RELEASE	Confirmed
	A-ABORT	Nonconfirmed
	A-P-ABORT	Provider-initiated
Connectionless	A-UNIT-DATA	Nonconfirmed

ACSE services have the following two modes of operation:

- *Normal mode:* Used in OSI management and TMN. In normal mode, ACSE services use presentation and session services, and A-ASSOCIATE services furnish the parameters of both presentation and session services. In normal mode, session services layer restrictions such as length must also be taken into consideration.
- *X.410-1984 mode:* Originally used with the MHS for ACSE. This mode uses a null presentation layer, and A-ASSOCIATE services do not provide presentation and session services parameters. There may be applications using the X.410-1984 mode, so it has been provided for backward compatibility with ACSE service of applications that require the X.410-1984 mode. This mode is not used in OSI management and TMN..

ACSE services subsequently explained assume underlying connection-oriented services. Connection-oriented ACSE services support the following three functional units:

- *Kernel functional unit:* Is always present and includes the services A-ASSOCIATE, A-RELEASE, A-ABORT, and A-P-ABORT.
- *Authentication functional unit:* Supports authentication while establishing association. Password verification is one of the commonly used methods for authentication. Authentication rules are negotiated while establishing the association. To support authentication, additional parameters are added to the A-ASSOCIATE and A-ABORT services. A-ASSOCIATE includes Authentication-mechanism name, Authentication value, and ACSE requirements. Authentication-mechanism name indicates the authentication function/mechanism used. If there is no authentication-mechanism name, then the authentication mechanism used is implicit. Authentication value includes the authentication value generated by the authentication function used. We will look into the parameter, ACSE requirements, later. A-ABORT has one additional parameter, diagnostic, included for authentication. Diagnostic includes the reasons related to authentication failures.
- *Application context negotiation functional unit:* Is used to negotiate application contexts while establishing association. If this functional unit is included, the acceptor has to select one or more application contexts mentioned in the Application Context Name or Application Context Name List. This functional unit is supported by adding the Application Context Name List and ACSE requirements parameters in the A-ASSOCIATE service. If the acceptor accepts the list of application contexts, then the list of application contexts is not included in the

response from the acceptor. However, if the list of application contexts are not acceptable to the acceptor, then the acceptor includes its own list of application contexts it can support.

7.4.1 ACSE Application Protocol Data Units (APDUS)

ACSE uses the following APDUs:

- *A-ASSOCIATE-REQUEST (AARQ):* A-ASSOCIATE.req is mapped to AARQ by the ACSE service provider.
- *A-ASSOCIATE-RESPONSE (AARE):* A-ASSOCIATE.rsp is converted to AARE.
- *A-RELEASE-REQUEST (RLRQ):* A-RELEASE.req becomes RLRQ.
- *A-RELEASE-RESPONSE (RLRE):* A-RELEASE.rsp is converted to RLRE.
- *A-ABORT (ABRT):* A-ABORT.req or A-P-ABORT.ind becomes ABRT. This and the preceding four APDUs are used with connection-oriented establishing and releasing of application associations in normal mode.
- *A-UNIT-DATA (AUDT):* A-UNIT-DATA.req is mapped to AUDT APDU and is used for connectionless application associations.

7.4.2 A-ASSOCIATE

During the connection establishment phase, each side must agree on how information will be exchanged. This phase is done by A-ASSOCIATE. It is during this phase that the application context and the rules for coordinating different ASEs are made. Negotiations on the functional units are also done. One of the functional units is the *negotiated release* functional unit. A-ASSOCIATE is a confirmed service, indicating that a reply is required.

A requestor sends its list of parameters and, on the other end, the acceptor may agree to the list furnished. Or, the acceptor may send its own list of rules for communicating. If the requestor can use this list, there is no problem. If the requestor cannot communicate with the list of the acceptor, it can send an ABORT.req.

In the following explanations, the terms *Calling, Called,* and *Responding* are used. *Calling* refers to the requestor of A-ASSOCIATE. *Called* corre-

sponds to the intended acceptor of A-ASSOCIATE, while *Responding* is the actual receiver of A-ASSOCIATE. This distinction between Called and Responding is made because the actual receiver and intended acceptor of A-ASSOCIATE can be different in some cases. For example, there can be an alternate receiver of A-ASSOCIATE if the original A-ASSOCIATE fails for some reason or if the called address is a generic one. A-ASSOCIATE parameters for the X.410-1984 mode are as follows:

- *Mode:* This can be X.410-1984. Normal mode is the default mode.
- *User Information:* This can be used by the requestor and acceptor to carry information to either end. One such example is forwarding passwords or keys to the other end. The AC may specify how this field is used and how the ASEs included are used in the AC.
- *Result:* The result of the association negotiation from the other (acceptor) side is provided in this field. It has three values: accepted, rejected (permanent), or rejected (transient). We have seen the action when an association and its terms are accepted. What happens when the other side is not able to accept the association and its associated terms? In this case, either *rejected (permanent)* or *rejected (transient)* is in the Result parameter, and the association is not established.
- *Result Source:* If an association is accepted by the acceptor, then this parameter has the value of the acceptor. On the other hand, if the association is rejected, then it will have the value of the service user, service provider, or presentation service provider.
- *Diagnostic:* This parameter can be used for providing diagnostic information, including specifically the reasons for rejecting an association.
- *Presentation Address (Calling, Called, and Responding):* These are respective presentation service access point (PSAP) addresses of the supporting presentation layer.
- *Quality of Service:* This indicates characteristics of services provided by the sessions layer, such as error rates, throughput, and others.
- *Session Requirements:* These provide for the negotiation of functional units.
- *Initial Synchronization Point Serial Number:* When we restart a session, it is resumed from this point.
- *Initial Assignment of Tokens:* This can be used for token assignments for purposes such as synchronization.
- *Session-Connection Identifier:* This is used to distinguish one connection from another.

In Normal mode, A-ASSOCIATE has the following additional parameters:

- *Application Context Name:* This is a mandatory parameter and specifies the AC name chosen by the requestor. The acceptor on the other end may return the same AC name, indicating that it agrees with the AC that will be used. The acceptor may also suggest a different AC. If the requestor agrees, then there is no problem; otherwise, the requestor will issue an A-ABORT.req.
- *AP Title (Calling, Called, and Responding):* This field has a string associated with an AP. Usually, AP titles are stored in an application title directory from which addressing information of the entities is retrieved using the AP title. The AP title can be either the directory name where AP titles are stored or an object identifier.
- *AP Invocation Identifier (Calling, Called, and Responding):* Each AP invocation is provided with a unique integer value to clearly differentiate one AP invocation from another AP invocation.
- *AE Qualifier (Calling, Called, and Responding):* Identifies an AE within an AP. The AE qualifier can be a relative distinguished name or an integer.
- *Presentation Context Definition List:* This provides abstract syntaxes and their presentation context identifier (PCI) values.
- *Presentation Context Definition Result List:* This refers to the abstract syntaxes and PCI values of the responding presentation service provider from the acceptor.
- *Default Presentation Context Name:* The abstract syntax name is provided for default conditions.
- *Default Presentation Context Result:* This is, again, the abstract syntax name of the responding service provider.
- *Presentation Requirements:* Functional units that may be used by the user of presentation services such as an AE are indicated in this parameter.
- *ACSE Requirements:* Used to indicate whether authentication or application context negotiation functional units are used. The ASN.1 for this parameter is a BIT STRING: A value of 0 indicates that an authentication functional unit is being used, and a value of 1 shows that an application context negotiation functional unit is being used. As mentioned earlier, the kernel functional unit is always present. If the ACSE requirements parameter is not present, then only the kernel functional unit is available.

In the preceding explanations, presentation parameters are governed by X.216, Presentation Service Definition. Similarly, X.215, Basic Connection-Oriented Service Definition, describes the session layer parameters. Let us examine further how the associations between two AEs are formed. These AEs can be between a manager and an agent or two managers acting as peers. A user of an ACSE sends an A-ASSOCIATE.req to the ACSE. This A-ASSOCIATE.req gets converted to AARQ and is wrapped with the headers of P-CONNECT. AARQ becomes user data in the P-CONNECT.req presentation protocol data unit (PPDU). This becomes user data for the S-CONNECT.req session protocol data unit (SPDU) in the session layer. In the transport layer, it becomes T-DATA.req. T-DATA.req, and similarly, other lower layers such as network, data link and physical layers, add their own layer headers. Then A-ASSOCIATE APDU is sent to the other end. In Figure 7-4, OSI lower layers refer to the layers below the session layer of the OSI seven-layer model. In other words, it includes layers such as the transport layer, network layer, data link layer, and physical layer. In Figure 7-4, there is no presentation layer for the X.410-1984 mode.

On the other end, another ACSE service user receives the frame as A-ASSOCIATE.ind. If the ACSE and AE are able to accept the terms of the association, the receiver of the ACSE forms A-ASSOCIATE.rsp, and the ACSE service provider converts to AARE and packages it with the header of P-CONNECT. Here we are making the assumption that we are in normal mode and are using presentation services. The AARE goes all the way down again, picking the necessary headers, and reaches the other end. The requesting ACSE service provider receives the AARE and converts it to A-ASSOCIATE.conf.

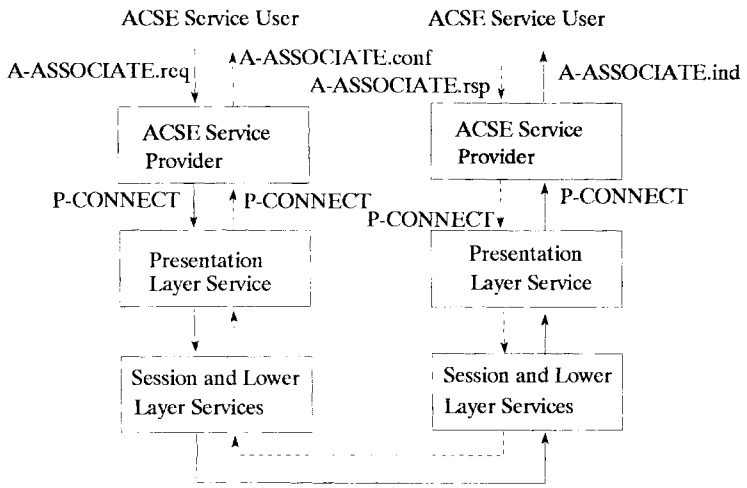


Figure 7-4 Functioning of A-ASSOCIATE (normal mode).

After having sent an A-ASSOCIATE.req, an AE cannot send another service request, such as another A-ASSOCIATE.req, to the same recipient. It can send only an A-ABORT.req to stop the association establishment procedures. The A-ABORT.req is turned into A-ABORT.ind on the other end of ACSE, and the association is, naturally, not established.

7.4.3 A-RELEASE

Once an association is established, obviously there is also a need to end the association when it is no longer needed. For this purpose, A-RELEASE is used. The key here is that no data or information in transit is lost. This is an orderly release. A-RELEASE can be sent from either the AE in a manager or the AE in an agent. This is a confirmed service, indicating that a reply is required. However, if a negotiated functional unit has been agreed upon during association, then the session layer takes over the release of the connection.

A-RELEASE has the following parameters:

- *Reason:* In A-RELEASE.req, this indicates how an association must be terminated—as normal, urgent, or user-defined. In turn, the acceptor on the other end states in the Reason parameter of A-RELEASE.rsp one of the three values: normal, not finished, or user-defined. The acceptor may indicate that this has been accepted or that there are data remaining, and there is flexibility on the part of an acceptor to define its own values for the user-defined values. This parameter is used in normal mode only.
- *User Information:* As is obvious from the name of the parameter, this consists of user data permitted within the rules set down by the AC and the ASEs used. This parameter is used in normal mode only.
- *Result:* This is a mandatory parameter in response and confirm. Result is used by the acceptor to indicate whether or not the acceptor has accepted the request to release the association.

The A-RELEASE function is illustrated in Figure 7-5. The first thing to remember is that A-RELEASE.req can be sent from either AE. The functioning of A-RELEASE is similar to that of A-ASSOCIATE. One major difference is that we use the RLRQ APDU as user data for the P-RELEASE PPDU. The responder service provider provides the RLRE APDU to the presentation layer below it. When an association is released, then lower-level presentation and session services connections are also released. Here also we assume normal mode operation.

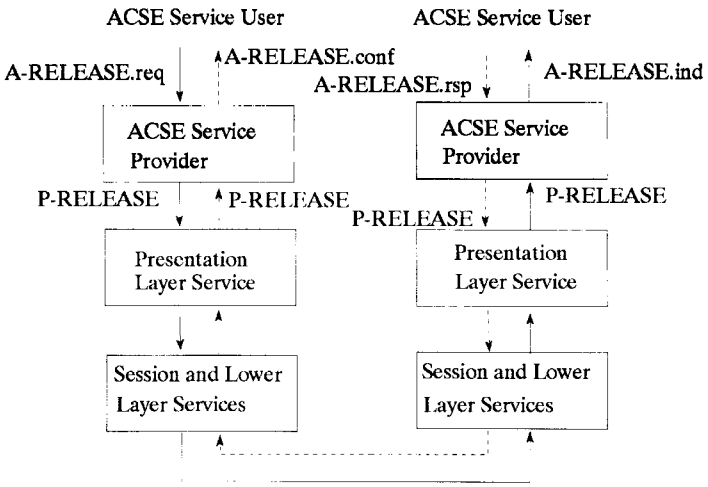


Figure 7-5 Functioning of A-RELEASE (normal mode).

7.4.4 A-ABORT

A-ABORT comes in handy when there have been errors, such as users no longer being able to communicate. The procedure followed by A-ABORT can be compared to a fire alarm drill: Leave everything as it is and rush to the fire exit! A-ABORT is a nonconfirmed service, indicating that a reply is not expected. The important points to note are that it is an abnormal release and that data or information in transit may be lost. Receiving an A-ABORT will stop everything, including data transfer related to the association. A-ABORT.req is sent by an AE if there are problems. Also, an ASE can send A-ABORT.ind to an AE. A-ABORT can be sent by the AEs in a manager or an agent. A-ABORT has the following parameters:

- *Abort Source:* This indicates who initiated the A-ABORT. It can be a service user (AE) or a service provider (ACSE). Abort Source is used only in normal mode and is mandatory in A-ABORT.ind.
- *User Information:* This has the provision to include user information by the acceptor of an A-ABORT service. In the original version of session service standards, user information could be only nine octets in length. However, if we use the addendum to the original version, Unlimited User Data, this restriction is not there and we can use any length to explain to the other end the reason for A-ABORT.

How the A-ABORT service functions is shown in Figure 7-6. The APDU used here is ABRT. The corresponding presentation service used here is P-U-ABORT. As we have seen, A-ABORT may be sent either by a

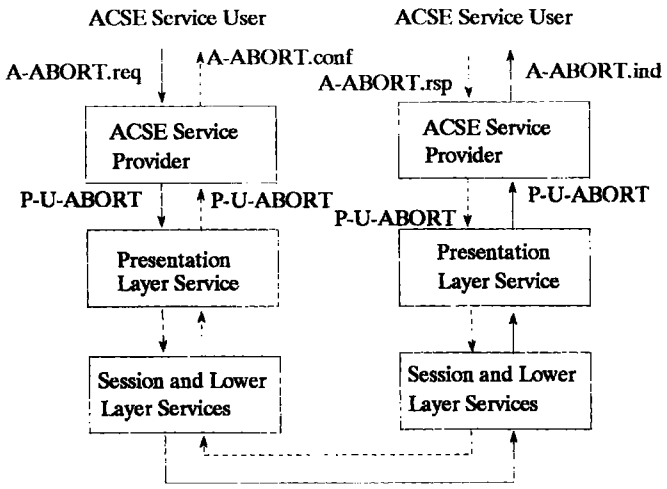


Figure 7-6 Functioning of A-ABORT (normal mode).

user of an ACSE service, which is an AE, or by the ACSE service provider itself.

7.4.5 A-P-ABORT

A-P-ABORT.ind is sent by an ACSE service provider to an AE. The errors involved may be the result of internal errors in an ACSE or errors in layers such as the presentation layer and other layers below it. In such a case, a service provider uses A-P-ABORT to terminate the association. As with A-ABORT, there is also a possibility of loss of information in transit. Reason is the only parameter associated with A-P-ABORT. Reason Included is related to presentation-layer reasons. Because the functioning of A-P-ABORT is similar to that of other ACSE services, we have not repeated the steps involved in A-P-ABORT. In the presentation layer, we use P-P-ABORT instead of P-U-ABORT as used in A-ABORT. Here, too, the APDU used is ABRT as in A-U-ABORT.

7.4.6 Connectionless ACSE

ACSE services have also been defined for the connectionless mode of transmission. The major difference is that, in the connectionless mode of transmission, the presentation and session layer parameters must be specified in each service primitive. In connection-oriented transmission, once a connection is established, we need not provide all the presentation and

session layer parameters all over again. We can reuse the knowledge of the connection already made.

Connectionless mode of ACSE has only A-UNIT-DATA service and uses the connectionless presentation service, P-UNIT-DATA. Connectionless mode of service supports only authentication functional unit. If authentication functional unit is included in the connectionless mode, then two additional parameters—authentication mechanism name and authentication value—are included. The parameter, ACSE requirements, is not required in A-UNIT-DATA, as the presence of the authentication mechanism name and authentication value parameters implicitly indicates the presence of the authentication functional unit.

The parameters included along with A-UNIT-DATA are Protocol Version, Application Context Name, AP Title (Calling and Called), AE Qualifier (Calling and Called), AP Invocation Identifier (Calling and Called), AE Invocation Identifier (Calling and Called), Implementation Information, and User Information. As we have already discussed these parameters in the connection-oriented ACSE, we will not discuss these parameters again.

While it is the normal practice for a manager to initiate an association with an agent, there is no rule that it should always be so. It is possible for an agent to initiate an association with a manager. Also, there is no restriction stating there should be only one association between a manager and an agent. There are cases where a manager and agent can have more than one association. It can also help balance the workloads. As an example, it may be a good idea to do Gets and process the Gets, say, for configuration management on one association, and to process the event reports due to threshold crossing for fault management on another association.

7.5 Remote Operations Service Element (ROSE)

An illustration best explains remote operations. In Figure 7-7, an application entity AE1 requests an operation to be performed on another application entity AE2. The outcome of the operation is reported back. The operation performed on AE2 is known as a *remote operation.* Here AE1 is known as the *invoker* and AE2 is termed the *performer.* ROSE primitives are listed in Table 7-2. Except for RO-REJECT, all are nonconfirmed services. ROSE services are analogous to *remote procedure calls* (RPCs). RPCs are sim-

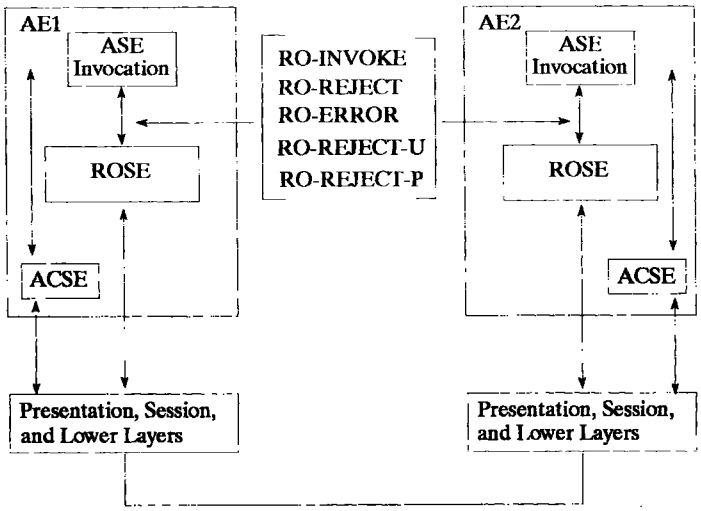


Figure 7-7
ROSE concepts.

ilar to the subroutine calls found in programming languages, except that RPC are made to processes in another system.

At first, an association is formed between the invoker (AE1) and performer (AE2) AEs, using the association service A-ASSOCIATE. After an association is formed for data transfer, ROSE interacts with the presentation service. These concepts are illustrated in Figure 7-7.

ROSE APDUs—ROIV, RORS, ROER, and RORJ—are formed in a *remote operation protocol machine* (ROPM), as shown in Figure 7-8. ROPM, just like CMIPM, is a finite-state machine. ROIV is used for invoking an operation on another system (performer). If there are no errors, then the results of the operation are conveyed back to the invoker using RORS. However, if there are errors, then the ROER APDU is used by the performer to report the errors to the invoker. If the invoker or performer cannot process an APDU, then the RORJ APDU is used. The correspondences between ROSE services and the APDU are provided in Table 7-3.

TABLE 7-2
Remote Operation Service Element (ROSE).

Services	Type
RO-INVOKE	Nonconfirmed
RO-RESULT	Nonconfirmed
RO-ERROR	Nonconfirmed
RO-REJECT-U	Nonconfirmed
RO-REJECT-P	Provider-initiated

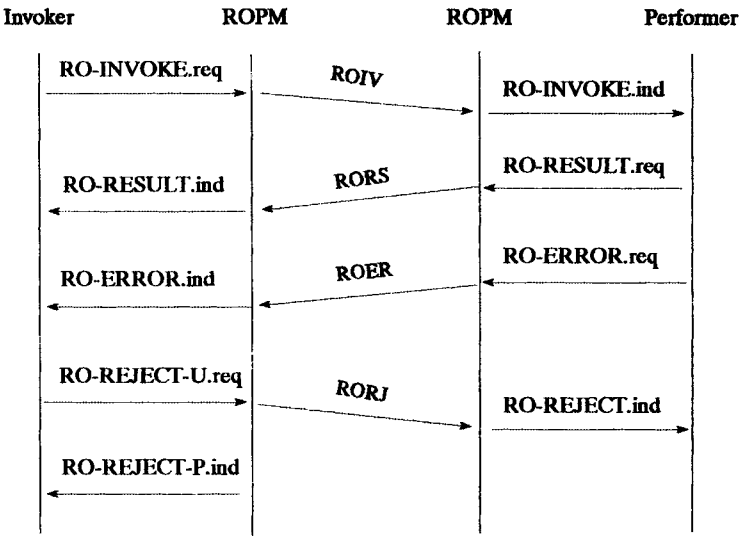


Figure 7-8
ROSE APDUs.

7.5.1 RO-INVOKE

The first service primitive of ROSE is *Invocation.* Here RO-INVOKE is used by AE1 to request that AE2 perform an operation. This operation may be a manager requesting an agent to create an object instance. The result of an operation by AE2 is transferred back to AE1 by RO-RESULT. RO-INVOKE has the following parameters:

- *Invoke-ID:* This is the identifier value of the RO-INVOKE.req and is a mandatory parameter.
- *Linked-ID:* This is an identifier of the parent invocation. If the ROSE invocation is child of another invocation, then this value is used.
- *Operation Value:* This is a mandatory parameter, containing the operation number, which is mutually agreed on between the invoker and the performer.

TABLE 7-3
Mapping ROSE Services and APDUs.

ROSE Services	APDU
RO-INVOKE.req/ind	ROIV
RO-RESULT.req/ind	RORS
RO-ERROR.req/ind	ROER
RO-REJECT-U.req/ind	RORJ
RO-REJECT-P.ind	RORJ

- *Argument:* This contains the argument for the operation stated in the Operation Value parameter.
- *Operation Class:* This can be synchronous or asynchronous operations. In synchronous operations, we wait for the reply to come before sending another operation, whereas in asynchronous operations we need not be concerned about receiving replies right away, and we can invoke other operations. ROSE has five class operations: Class 1 (synchronous with replies involving results and errors), Class 2 (asynchronous with results and errors), Class 3 (asynchronous with only errors), Class 4 (asynchronous with only results), and Class 5 (asynchronous with no results or errors).
- *Priority:* This states the priority of the invocation. Smaller numbers indicate a higher priority.

7.5.2 RO-RESULT

The result of a successful operation of a performer is included in the RO-RESULT.req parameter. Parameters used are Invoke-ID, Result, Operation Value, and Priority. Here Invoke-ID is the identifier of RO-RESULT.req and is the only mandatory parameter. If the Result parameter is present, then the Operation Value parameter is also present.

7.5.3 RO-ERROR

There is always the possibility of error while performing some processing. In the application entity AE2, if there is an error during performance of an operation, RO-ERROR is used to report the errors. RO-ERROR's parameters are Invoke-ID, Error Value, Error, and Priority. Of these four parameters, Invoke-ID and Error Value are mandatory parameters. Here, too, Invoke-ID is the invoke identifier of RO-INVOKE.req for which the error is being reported. Error data is placed in the Error parameter, and it is given a value that is placed in Error Value.

7.5.4 RO-REJECT

If there are problems in accepting an invocation as a result of error by the user of ROSE, such as AE1 or AE2, then RO-REJECT-U is used. Here the

user includes the Invoke-ID of the ROSE service from which the RO-REJECT-U has been generated. Invoke-ID is a mandatory parameter. Another parameter is Problem, which is mandatory and can include Invoke Problem, Return Result Problem, or Return Error Problem parameters. Priority is an optional parameter.

RO-REJECT-P is used to inform a ROSE user if a problem is detected in the ROSE service provider. The Invoke-ID is that of the invocation which encounters the problem. This may be due to unrecognized fields, unacceptable structure, or badly structured PDUs. It may be noted that these problems can be in either the ROSE invoker or the ROSE performer.

ROSE protocols are described in X.219, Remote Operations: Model, Notation, and Service Definition (Reference 7.7) and X.229, Connection-mode Protocol Specifications (Reference 7.8).

7.6 Common Management Information Service Element (CMISE)

The common management information service element (CMISE) provides the management operations and notification used by CMIP and services defined by CMIS. The services provided by CMISE are shown in Table 7-4. Management information may be in the form of operations or notifications. Those processes that use these services are known as CMISE service users. The CMISE service provider provides the CMISE services.

TABLE 7-4

Common Management Information Service Element (CMISE) Services.

Service	Type
M-GET	Confirmed
M-SET	Confirmed/nonconfirmed
M-CREATE	Confirmed
M-DELETE	Confirmed
M-CANCEL-GET	Confirmed
M-EVENT-REPORT	Confirmed/nonconfirmed
M-ACTION	Confirmed/nonconfirmed

From Table 7-4, we notice that M-GET, M-CREATE, M-CANCEL-GET, and M-DELETE are confirmed services. That means the responses have to be sent when these services are used. The remaining CMISE services such as M-EVENT-REPORT, M-SET, and M-ACTION can be either confirmed or nonconfirmed. The implication of a nonconfirmed service is that no reply is sent. However, the confirmed responses can be linked together if required. As an example, if a scope request selects multiple managed objects, then the response consists of responses from each of the managed objects selected. The Linked Identifier parameter is used to assemble the responses by the requestor of CMISE services. The prefix M- used in CMISE services stands for management, and it indicates that these services are used for management-related operations and notifications.

For establishing associations prior to the use of CMISE services, A-ASSOCIATE is used. The initiating CMISE service user includes the CMIS-specific parameters Functional Units, Access Control, and User Information in the User Information parameter of A-ASSOCIATE. Here, additional functional units that can be supported are negotiated. Additional functional units refers to the functional units related to CMIS; these functional units are explained in Section 7.7. If a Functional Unit parameter is not mentioned, then only the kernel functional unit is supported. The set of additional functional units, if agreed on by both the user and the provider, will be used for management operations and notifications.

Access Control establishes all the access control privileges for associations. However, if Access Control is mentioned as a parameter in a service request, it will be valid only for the invocation and not for other management operations. Any user information that must be exchanged between a user and a provider can be included in the User Information parameter.

Finally, when connections are formed there should also be a means to break them. Usually, A-RELEASE is used for releasing the association between a user and a provider. However, when the situation is hopeless and nothing else works, then A-ABORT can be used. A CMISE service user or CMISE service provider can use A-ABORT. The A-ABORT user information parameter includes the Abort Source and User Information parameters. The Abort Source parameter indicates whether the CMISE service user or CMISE service provider initiated the A-ABORT. Context-specific user information, if any, is included in the user information by service user or service provider that initiates the abort.

Some parameters are common to all the CMISE services (see Table 7-5). Some of these parameters are required, but the use of some is optional.

TABLE 7-5

CMISE Services and Parameters.

Services	Parameters
M-EVENT-REPORT.req/ind	Invoke-ID (M), Mode (M), Managed Object Class (M), Managed Object Instance (M), Event Type (M), Event Time, Event Information
M-EVENT-REPORT.rsp/conf	Invoke-ID (M), Managed Object Class, Managed Object Instance, Event Type, Current Time, Event Reply, Errors
M-GET.req/ind	Invoke-ID (M), Base Object Class (M), Base Object Instance (M), Scope, Filter, Access Control, Synchronization, Attribute Identifier List
M-GET.rsp/conf	Invoke-ID (M), Linked-ID, Managed Object Class, Managed Object Instance, Current Time, Attribute List, Errors
M-CANCEL-GET.req/ind	Invoke-ID (M), Get Invoke-ID (M)
M-CANCEL-GET.rsp/conf	Invoke-ID (M), Errors
M-SET.req/ind	Invoke-ID (M), Mode (M), Base Object Class (M), Base Object Instance (M), Scope, Filter, Access Control, Synchronization, Modification List
M-SET.rsp/conf	Invoke-ID (M), Linked-ID (M), Managed Object Class, Managed Object Instance, Attribute List, Current Time, Errors
M-ACTION.req/ind	Invoke-ID (M), Mode (M), Base Object Class (M), Base Object Instance (M), Scope, Filter, Access Control, Synchronization, Action Type, Action Information
M-ACTION.rsp/conf	Invoke-ID (M), Linked-ID, Managed Object Class, Managed Object Instance, Action Type, Current Time, Action Reply, Errors
M-CREATE.req/ind	Invoke-ID (M), Managed Object Class (M), Managed Object Instance, Superior Object Instance, Access Control, Reference Object Instance, Attribute List
M-CREATE.rsp/conf	Invoke-ID (M), Managed Object Class, Managed Object Instance, Attribute List, Current Time, Errors
M-DELETE.req/ind	Invoke-ID (M), Base Object Class (M), Base Object Instance (M), Scope, Filter, Access Control, Synchronization
M-DELETE.rsp/conf	Invoke-ID (M), Linked-ID, Managed Object Class, Managed Object Instance, Current Time, Errors

M, mandatory parameter (for parameters column only).

Invoke-ID is one of the required parameters. Each CMIS operation is given a unique identifier to distinguish one CMIS operation from the other. This parameter is necessary because many CMIS operations or only one particular kind of operation can be performed. Whenever any management operation or notification is done, there is always a possibility of errors. So Errors is a conditional parameter used when sending a response or confirmation.

Another CMISE parameter is Managed Object Class. It identifies the managed object class on which an operation is performed. Managed Object Instance refers to a specific object instance in a possible set of instances of a managed object class. These two parameters are not included in M-CANCEL-GET. Table 7-5 lists the parameters of the CMISE services.

7.6.1 M-EVENT-REPORT

When an object instance has something to report, such as a change of state or errors, then M-EVENT-REPORT is used. Naturally, the parameters must include various details, such as the type of information, when the information was generated, when notification occurred, when response to a notification was generated, and details of the information.

M-EVENT-REPORT is important and is widely used for reporting notifications emitted by managed objects. Managed objects can generate notifications, which the agents send as an M-EVENT-REPORT to managers.

When an M-EVENT-REPORT is sent, there is a possibility of errors, which are included in M-EVENT-REPORT.conf. These are included in the Errors service parameter. Figure 7-9 illustrates how M-EVENT-REPORT is used. In the figure, *invoking* and *performing service users* refer to the services, which may be a process in an agent or a manager. Because there is no response in the nonconfirmed mode, the figure is applicable only to the confirmed mode.

7.6.2 M-GET

It is possible to retrieve values from managed objects. For this purpose, the M-GET CMIS primitive is used. It is a confirmed service and can be canceled by an M-CANCEL-GET service. Here, too, the parameters supported must be able to retrieve the information from the object instances.

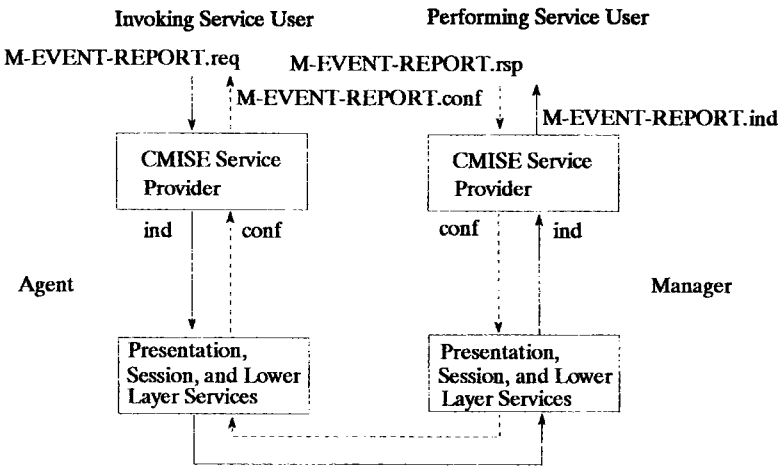


Figure 7-9 Working of M-EVENT-REPORT.

There may be more than one object instance, so the parameters of scoping and filter are used to select a group of managed objects. In such a case, there is also the possibility of more than one reply. To handle this situation, the Linked-ID parameter is used. Figure 7-10 depicts M-GET without linked replies. Figure 7-11 illustrates the functioning of M-GET if linked replies are used. The synchronization parameter indicates how to retrieve the values. When an object instance value is retrieved, it may be important to limit the access to managed objects. For this purpose, the Access Control parameter is used. An example of how the M-GET functions is shown in Figure 7-12.

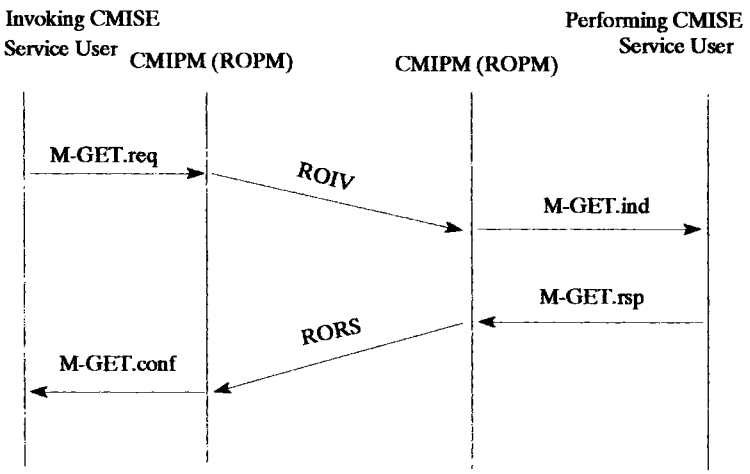


Figure 7-10 Functioning of M-GET without linked replies.

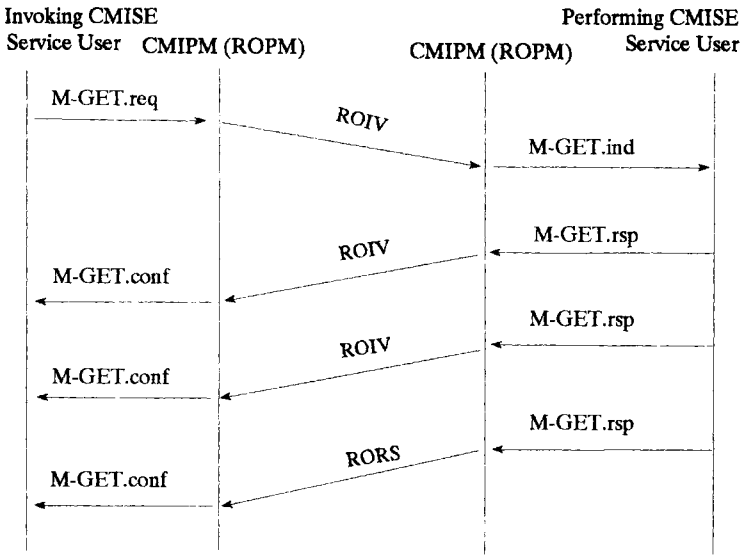


Figure 7-11 Functioning of M-GET with linked replies.

7.6.3 M-CANCEL-GET

As mentioned earlier, M-CANCEL-GET is used to abort any M-GET service that is pending. This service is useful when the amount of information being returned is unexpectedly large. The M-GET service may lead to long multiple replies. Besides, most of the information may be useless and, in some cases, it may be a duplicate of information already received. M-CANCEL-GET is very useful in such cases. Here, Get Invoke-ID refers to the identifier of the M-GET operation to be canceled.

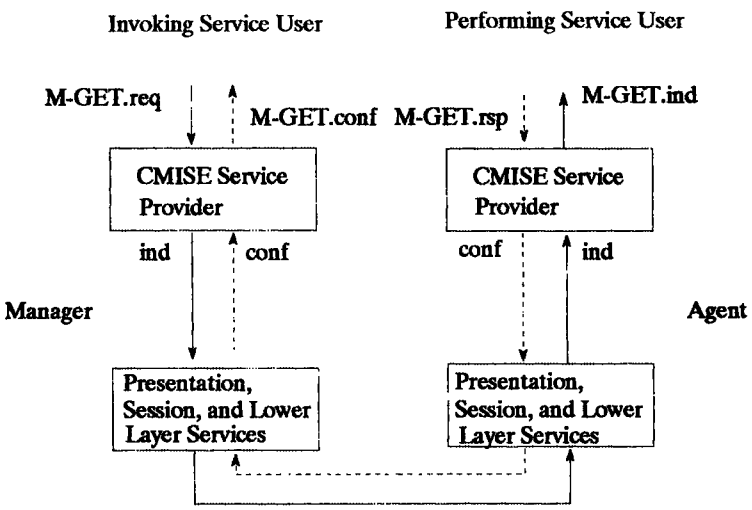


Figure 7-12 Functioning of M-GET.

7.6.4 M-SET

The attribute values of managed objects may need to be changed to control the resource represented by a managed object. For this, the M-SET CMISE service is used. It has parameters such as Linked-ID, Scoping, Filter, Synchronization, and Access Control to control the operations performed on an object instance or instances. The parameter list is similar to that of M-GET. While M-GET is used to retrieve the values of the object instances, M-SET is used to modify the values that have been assigned. M-SET can be a confirmed or nonconfirmed service.

7.6.5 M-ACTION

M-ACTION is used to perform management action on managed objects. This operation may be confusing. Why do we need it, and where do we use it? Let us look at a special case in which some operation such as EXECUTE must be performed on an object instance or a set of object instances. In this case, EXECUTE can be packed in M-ACTION in Action Type and Action Information parameters. Except for M-ACTION, there is no other CMISE service that can be used to carry the operation EXECUTE. Hence, M-ACTION provides the escape mechanism to accommodate any operation that must be defined for a managed object class but that cannot be accurately modeled with a standardized attribute-oriented operation, M-CREATE, or M-DELETE.

When defining managed objects, we must clearly spell out the operations that can be performed on them. M-ACTION, like M-GET and M-SET, has parameters to identify a set of object instances and how they can be handled by the Scoping, Filter, Synchronization, and Access Control parameters. They also have linked identifiers to receive multiple replies from more than one object instance. The definer of M-ACTION can specify request/response syntaxes, the number of responses, and any other behavior required.

7.6.6 M-CREATE

When managed object classes are defined, instances of managed object classes must be created. For this purpose, the M-CREATE CMISE service is used. Managed objects that are created should subsequently be clearly identifiable. Thus, the reply should contain attribute values, the managed object class identifier that is the registered value, details on how to access

the object instance, the superior object instance, and other details. The parameters of the superior object instance are optional, because the object instance's superior is obvious from that object's distinguished name.

7.6.7 M-DELETE

The reverse action of Create is Delete. For this purpose, the M-DELETE CMISE service is provided; it is a confirmed service. M-DELETE has parameters to do the scoping and filtering to select a set of object instances. The method of deleting these object instances is furnished by the Synchronization parameter. Using M-DELETE deregisters the object instance or instances involved. Some of the parameters, such as managed object classes and object class instances, are similar to those of the M-CREATE CMISE service.

7.7 Functional Units

For negotiating the CMISE services during association establishment, CMISE services are grouped into different functional units, which makes it easy to negotiate which services are to be included. Let us examine each of the functional units.

7.7.1 Kernel Functional Unit

The kernel functional unit provides the basic CMISE services such as M-EVENT-REPORT, M-GET, M-SET, M-ACTION, M-CREATE, and M-DELETE. However, it does not include specialized functions such as multiple replies, scoping, filtering, and synchronization capabilities. The parameters relevant to these services are included if functional units providing the specialized functions are included.

7.7.2 Multiple Object Selection Functional Unit

The multiple object selection functional unit is used for specifying the scoping and synchronization capabilities. These two capabilities are not

applicable to M-EVENT-REPORT and M-CREATE. When this functional unit is specified, then the multiple reply functional unit is also required.

7.7.3 Multiple Reply Functional Unit

The multiple reply functional unit makes use of the Linked-ID parameter in different CMISE services. This parameter is not used in M-EVENT-REPORT and M-CREATE. Combining multiple replies is a powerful concept and is very useful when a large amount of data has to be sent in replies. Some common scenarios for obtaining a large amount of data in replies occur during operations involving multiple managed objects or due to M-ACTION, which can result in a large amount of data. These large, continuous streams of replies may sometimes pose problems, in which case M-CANCEL-GET can be used to stop the further flow of data.

7.7.4 Filter Functional Unit

The filter functional unit makes use of the Filter parameter. As we have seen earlier, the Filter operation can be used along with scoping to select managed objects. The Filter parameter is not available in M-EVENT-REPORT and M-CREATE.

7.7.5 Extended Service Functional Unit

Availability of P-DATA is assumed for the CMISE services. To make use of the additional presentation layer services, the extended service functional unit was anticipated. However, no one has attempted to standardize any use of this functional unit.

7.7.6 Cancel Get Functional Unit

We have seen earlier that the kernel functional unit does not include M-CANCEL-GET CMISE service. So, to provide M-CANCEL-GET service, we must include this functional unit at the time of negotiating the establishment of the association.

7.8 Common Management Information Protocol (CMIP)

CMIS primitives (e.g., M-GET.req) are used by the CMISE users. These CMIS primitives are conceptually converted to CMIP PDUs in CMIPM, as shown in Figure 7-13. These PDUs are used to exchange management information between two peer open systems. CMIP operations, which are part of the CMIP PDUs, are listed in Table 7-6.

Basically, CMIPM converts a CMIS primitive to the equivalent CMIP PDU and uses one of the appropriate ROSE PDUs, such as RO-INVOKE, RO-RESULT, or RO-ERROR, to transfer the CMIP PDU to the lower presentation layer. As an example, CMIS M-GET.req maps to the CMIP m-Get operation, and this m-Get operation becomes a part of the RO-INVOKE PDU. It will be necessary to envelop this RO-INVOKE PDU with the parameters of P-DATA of presentation services.

On the other end, CMIPM also converts the PDU containing CMIP operation to a CMIS primitive and passes the CMIS primitive to the CMISE service user. One example of the conversion is that the CMIP m-EventReport operation gets mapped to the CMIS M-EVENT-REPORT.ind. Note that the terms CMIPM or ROPM can be misleading because of the M. These are mainly FSMs, which can be software modules or microcode, and the term *machine* should be not interpreted beyond this in its scope.

CMIS is explained in X.710, Common Management Information Service Definition for CCITT Applications (Reference 7.11), and CMIP is

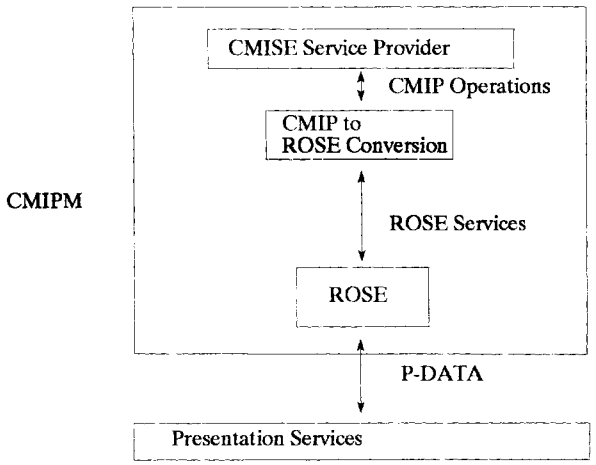


Figure 7-13
Relationship between CMIPM and presentation services.

TABLE 7-6

Mapping of CMISE Services and CMIP Operations.

CMISE Services	CMIP Operation	Notes
M-EVENT-REPORT	m-EventReport	—
M-GET	m-Get	m-Linked-Reply for rsp/conf.
M-CANCEL-GET	m-Cancel-Get	—
M-SET	m-Set	m-Linked-Reply for rsp/conf.
M-ACTION	m-Action	m-Linked-Reply for rsp/conf.
M-CREATE	m-Create	—
M-DELETE	m-Delete	m-Linked-Reply for rsp/conf.

defined in X.711, Common Management Information Protocol Specification for CCITT Applications (Reference 7.12). Note that the X.710 and X.711 standards only mention connection-oriented management data transfers between a manager and agent.

7.9 Systems Management Operations on Objects and Attributes

There are two kinds of operations on managed objects. One kind can be done on attributes; the other can be on managed objects of a managed object class. These operations are shown in Figure 7-14. Get, Replace Attribute Value, Replace With Default Value, Add Member, and Remove Member operations can be done on attributes. These are used as explained in the following:

- For retrieving values of attributes of a managed object, a Get operation can be done.
- To set the values of one or more attributes of a managed object to specified values, a Replace Attribute Value operation is done.
- If the value or values of attributes are to be set to default values, a Replace With Default Value operation is used.
- For adding or removing members in the case of set-valued attributes, an Add or Remove Member operation can be used.

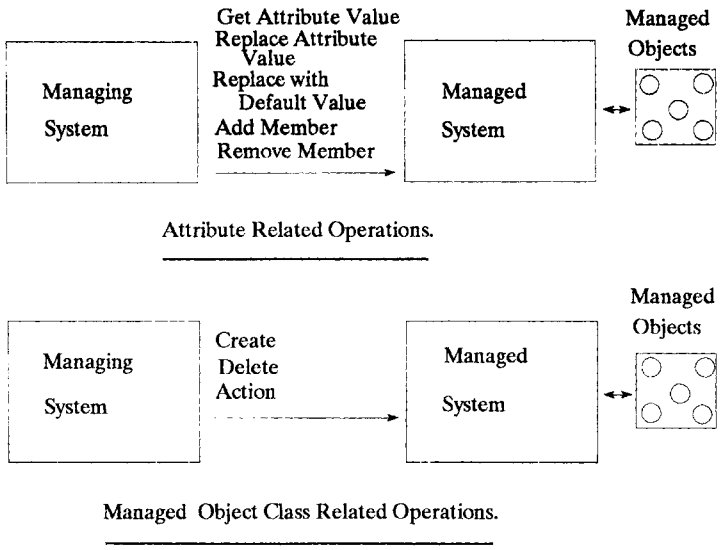


Figure 7-14
Managed object and attribute-related mangement operations.

When defining a managed object class, the operations that can be done by a managed system on managed objects must be included and the semantics of each operation must be described. Delete and Action are operations that can be performed on a managed object, as explained in the following:

- If an object instance can be created by a management operation, then a Create operation of an instance of the specified managed object class must be supported by the managed system for at least one name binding.
- If an object instance is to be deleted, then a Delete operation is done. However, it is not necessary that all managed objects support this. Here also, support for Delete is specified in the name binding.
- The Action operation is defined for use when none of the preceding operations can be done. If we want to perform operations such as EXECUTE on a managed object class, Action can be used. These operations are packaged in the Action operation.

7.10 Pass-through Services

Pass-through services (see Figure 7-15) can be used by layer management, (N) layer operations, or those cases that do not cover the OSI. Pass-through

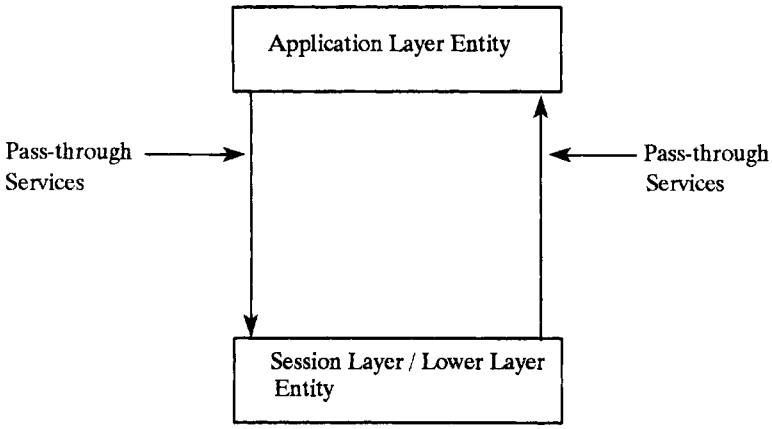


Figure 7-15
Pass-through services.

services can be mapped to management operations as shown in Table 7-7. Pass-through services provide the following:

- *Object-related operations:* Managed objects can be created and deleted, and actions can be performed on managed objects. This is done by using the primitives PT-CREATE, PT-DELETE, and PT-ACTION, respectively. Here, PT stands for pass-through services.
- *Attribute-related operations:* These include changing the value of one or more attributes to default values using PT-SET, reading the values of attributes via PT-GET, replacement of the value of one or more attributes with the help of PT-SET, and addition or removal of one or more members of sets using PT-SET.
- *Notifications:* These are sent for changes in attributes or managed objects. For this, PT-EVENT-REPORT is used.

TABLE 7-7
Mapping of Pass-through Services and Management Operations.

Pass-through Services	Management Operations
PT-CREATE	Create
PT-DELETE	Delete
PT-ACTION	Action
PT-SET	Replace, Add, Remove, and Replace with Default
PT-GET	Get
PT-EVENT-REPORT	Notification

7.11 Summary

In this chapter, we have discussed what ACSE, ROSE, CMISE, and CMIP mean. We have examined these important ITU-T functions from the perspective of TMN and systems management and have discussed how they can be grouped to support TMN and systems management functions. We have also briefly discussed the topics of pass-through services.

7.12 References

ACSE

7.1. ITU-T Recommendation X.217 (ISO 8649), Information Processing Systems, Open Systems Interconnection, Service Definition for the Association Control Service Element, 1995.

7.2. ITU-T Recommendation X.217 Amendment 1 (ISO 8649 AM 1), Information Processing Systems, Open Systems Interconnection, Service Definition for the Association Control Service Element, Amendment 1: Support of Authentication Mechanisms for the Connectionless Mode, 1996.

7.3. ITU-T Recommendation X.227 (ISO 8650-1), Information Processing Systems, Open Systems Interconnection, Connection-oriented Protocol for the Association Control Service Element: Protocol Specification, 1995.

7.4. ITU-T Recommendation X.227 Amendment 1 (ISO 8650-1 AM 1), Information Processing Systems, Open Systems Interconnection, Connection-oriented Protocol for the Association Control Service Element: Protocol Specification. Amendment 1: Incorporation of Extensibility Markers, 1996.

7.5. ITU-T Recommendation X.237 (ISO 10035-1), Information Processing Systems, Open Systems Interconnection, Connectionless Protocol for the Association Control Service Element: Protocol Specification, 1995.

7.6. ITU-T Recommendation X.237 Amendment 1 (ISO 10035-1 Amd. 1), Information Processing Systems, Open Systems Interconnection, Connectionless Protocol for the Association Control Service Element: Protocol Specification. Amendment 1: Incorporation of Extensibility Markers and Authentication Parameters, 1996.

ROSE

7.7. ITU-T Recommendation X.219 (ISO 9072-1), Open Systems Interconnection, Service Definitions, Remote Operations: Model, Notation, and Service Definition, 1988.

7.8. ITU-T Recommendation X.229 (ISO 9072-2), Open Systems Interconnection, Connection-mode Protocol Specifications, 1988.

RTSE

7.9. ITU-T Recommendation X.218 ((ISO 9066-1), Open Systems Interconnection—General Service Definitions, Reliable Transfer: Model and Service Definition, 1993.

7.10. ITU-T Recommendation X.228 (ISO 9066-2), Open Systems Interconnection, PICS Proformas, Reliable Transfer: Protocol Specification, 1988.

CMIS AND CMIP

7.11. ITU-T Recommendation X.710 (ISO 9595), Information Technology, Open Systems Interconnection, Common Management Information Service Definition for CCITT Applications, 1991.

7.12. ITU-T Recommendation X.711 (ISO 9596-1), Information Technology, Open Systems Interconnection, Common Management Information Protocol Specification for CCITT Applications, 1991.

7.13 Further Reading

ITU-T Recommendation, X.650, Information Technology, Open Systems Interconnection, Basic Reference Model, Naming and Addressing, 1996.

ITU-T Recommendation, X.660, Information Technology, Open Systems Interconnection, Procedures for the Operation of OSI Registration Authorities, General Procedures, 1992.

ITU-T Recommendation, X.660, Amendment 1, Information Technology, Open Systems Interconnection, Procedures for the Operation of OSI Registration Authorities, General Procedures, Incorporation of Object Identifiers Components, 1996.

This page intentionally left blank.

CHAPTER 8

Internet Network Management: SNMPv1, SNMPv2, and SNMPv3

8.1 Introduction

In the Internet arena, the management of networks, devices, and hosts is referred to as *network management*, in contrast to the terms used in TMN. To closely reflect the protocols and to be consistent with the terms used in the Internet community, we prefer to retain the term *network management* rather than *systems management*.

Originally, Internet network management was done using the *simple gateway monitoring protocol* (SGMP). Then the *simple network management protocol* (SNMP) was defined for the management of networks and network devices. However, the syntax and semantics of SNMP are different from those of SGMP. The protocol stack of SNMP is shown in Figure 8-1.

The objects to be managed are defined in the *management information base* (MIB). These objects must follow a certain set of rules as mentioned in the *structure of management information* (SMI) such that an object defined by the *X* group is compatible with the definition of the object by the *Y* group.

Note that objects in the Internet and those in TMN are different. Internet objects are similar to attributes in a TMN managed object, and an Internet object group can best be described as analogous to a TMN managed object class. In the Internet, an object is more like a variable found in programming languages; it has a syntax and semantics. Each object can have one or more object instances, each of which, in turn, has one or more values. In the following text, the terms *object* and *variable* are used interchangeably because of the closeness in the semantics of the two terms.

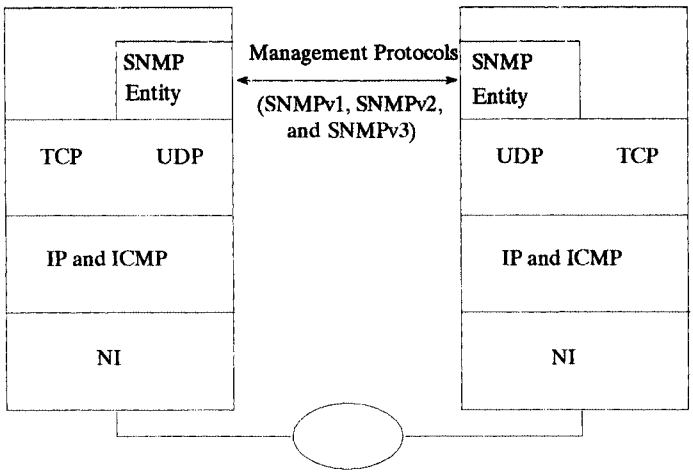


Figure 8-1
SNMP management protocols.

The documentation on SNMP is vast. In addition to SNMPv1 we have different versions of SNMPv2, and SNMPv3 has been introduced. So in this chapter we will cover only the important points regarding each of these SNMP versions. For details, readers should look into the Requests for Comments (RFCs) mentioned in the references.

The network management system consists of one or more *network management stations* (NMSs) and one or more *network elements* (NEs) with network management functions, management information, and management protocols. The network management system is shown in Figure 8-2. An NMS is analogous to a manager in concept, and the network management functions of NEs are rolled into agents. NMSs have the following primary functions:

- Retrieve values of objects in NEs using agents. This management operation is done by a GetRequest. A GetReguest is similar to a Get of CMIP.
- Change or alter the values of objects in NEs using agents. This can be done by a SetRequest. A SetRequest is similar to a Set of CMIP.
- Process the notifications or traps received from NEs.

Network management functions are responsible for monitoring and controlling the managed nodes. SNMP provides the management protocol for network management in the Internet world. The application entities, which can be managers or agents, exchange management information. These application entities are known as SNMP application entities. An NMS sends requests and receives responses from the agents. The NMS can also receive asynchronous reports on events, known as *traps*. A trap is an

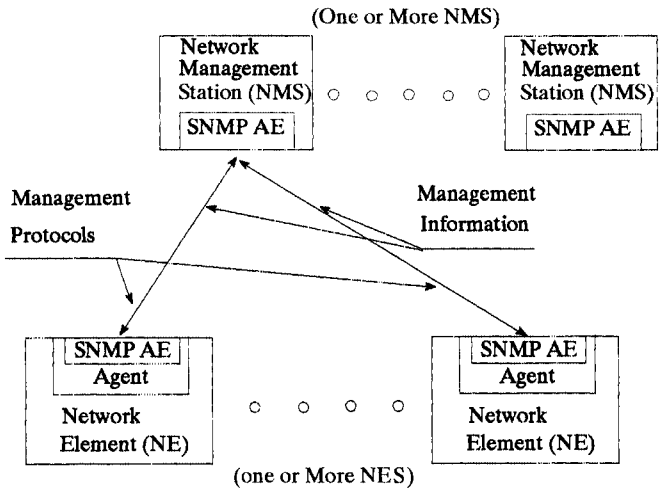


Figure 8-2 Internet network management.

unsolicited and unconfirmed message sent from an agent. Agents reside in NEs, which are also sometimes referred to as *managed nodes*. These managed nodes are devices such as hosts, routers, LAN adapters, modems, multiplexers, hubs, and printers. Agents have the following important functions:

- They are instrumented to retrieve management information from the objects, which represent the resources.
- They alter the values of objects.
- They receive responses and traps and send these, in turn, to network management stations.

SNMP is used to communicate management information between a management station and the agents. The management applications are built on the top of network management stations. Some key philosophies involved in the design of SNMP are as follows:

- The management protocol should be as simple as possible and must be provided at the lowest cost. The management protocols should have bare minimum operations.
- Network management must be robust and able to provide services, even when the state of the network is not quite reliable or when there are too many errors. SNMP must be able to handle the worst-case scenarios.
- The standard MIB will have core objects defined, and new objects to cover managed devices or managed nodes should be added to the MIB as needed. This extensibility should be easy.
- In the workload distribution between the NMS and the agents in the managed nodes, the major workload will be off-loaded to the NMS, such that it will be easy to add new managed nodes to the network. In other words, SNMP uses the *minimalist* approach. As per this approach, most of the network management functions are centralized in an NMS and the agents are kept simple. Reducing the workload in agents minimizes the need for processing power in the devices. This will also aid in the easy extensibility of the managed nodes supported.
- The transport used for carrying management protocols is connectionless User Datagram Protocol (UDP). It is not necessary to have the facilities provided by a connection-oriented transport, because they add complexity to the transport.
- The initial objective was to ensure easy migration to the OSI/ITU-T network management. Now, due to the popularity of SNMP, this objective is in question. Some proponents of SNMP support abandon-

ing the ultimate migration to OSI/ITU-T and instead favor the independent growth of Internet network management.

8.2 Internet Network Management Framework (SNMPv1)

The original network management framework consisted of RFC 1155 (Reference 8.1), RFC 1157 (Reference 8.2), and RFC 1212 (Reference 8.3). This is known as SNMP Version 1 (SNMPv1) (see Figure 8-3). The syntax and semantics used for the definition of objects for network management are furnished in RFC 1155, and are known as the structure of management information (SMI). SMI primarily states how to define objects and how to access them. RFC 1157 furnishes the SNMP management protocol used for accessing the objects. This management protocol is used for monitoring and controlling objects. RFC 1212 furnishes guidelines for defining new MIB modules. RFC 1213 (Reference 8.4) furnishes the definitions of a core set of objects that are known as MIB-II and that provide the base set for network management.

SNMP needs the support of the transport and network layers below it. The transport layer provides multiplexing and demultiplexing of services. This can result in many-to-many relationships that are possible between SNMP entities. Also, transport protocols provide for the end-to-end checksum, thus improving the reliability of data transfer.

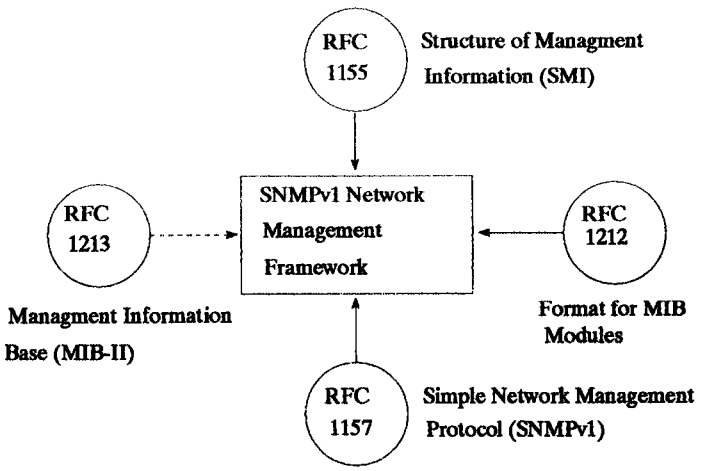


Figure 8-3 SNMPv1 network management framework.

The network layer provides for the routing capabilities between networks. Also, this layer shields the SNMP entities from differences in the media. In addition, the fragmentation and reassembly of packets of different sizes transmitted across the networks are provided by this layer. However, to reduce the possibility of fragments being lost, it is better to send small packets.

In some cases, it may not be necessary to use the transport and network layers. The network management functions may be provided directly over data link layers, as in the case of network management for point-to-point or out-of-band (OOB) management directly to the managed devices. In OOB, connections are sometimes made from management stations to the managed devices directly via dial-ups.

However, sometimes the network layer will be required. When management must be done over devices across networks, then it may be necessary to use the routing capabilities of the network layers. Skinny UDP/IP stacks may also be required in some cases. So, ultimately, these decisions must be made while designing management functions for networks.

Because much overhead is involved in maintaining one or more connections or establishing a connection and tearing it down for each SNMP entity operation on an object, a connectionless transport layer is preferred. This is one of the reasons for choosing UDP. UDP provides unreliable datagram service as a trade-off to keep the protocol simple. Every piece of management information is supposed to be carried in a single and independent transport datagram. This way, the complexity involved in assembling frames and in recovery after failures, such as frames not being received in order or lost frames being retransmitted, is avoided.

Internet network management operations are kept simple by limiting the operations to retrieve the value of a variable or set the value of a variable. Also, the traps sent from agents are very limited so as to reduce network traffic. Imperative commands, or commands that trigger some other action, are avoided. The Internet network management functions can be part of the network operations center (NOC). The NOC is usually a central location for monitoring the operation of a network and rectifying network problems.

The exchange of management information between an NMS and an NE is done by SNMPs. Each NE has a set of objects also known as the SNMP MIB view. We will discuss the SNMP MIB in detail in a later section. When an SNMP AE in an agent is associated with a set of SNMP AEs in one or more managers, this pairing is known as a *community* (Figure 8-4). Each com-

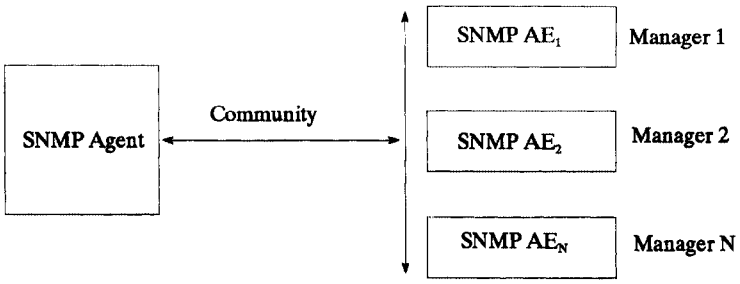


Figure 8-4
SNMPv1 community.

munity has an identifier known as the *community name*. The data type of the community name is OCTET STRING of a size from 0 to 255 octets. Only printable ASCII characters are allowed.

An SNMP message consists of a version identifier, an SNMP community name, and an SNMP PDU. The messages exchanged between an NMS and agents are independent of each other. Assume that we are keeping count of messages sent from an NMS to an agent. We have sent 15 messages, and message 16 is sent from the management station to the agent. Now, message 16 does not have any dependency on message 15. Also, in the SNMP implementation, it is recommended that message length not exceed 484 octets.

The SNMP community authentication scheme, defined by a set of rules, determines whether the messages sent between AEs are authentic. One example is that of an SNMP agent. It is possible that this agent would not send a trap to one particular manager, in which case checking whether the agent is really allowed to send the trap must be done by the authentication scheme.

8.3 Internet Objects

SNMP uses ASN.1 for defining objects and PDUs exchanged by the management protocol. However, all data types defined in ITU-T ASN.1 are not used. It was thought that by using a subset of data types, other data types could be constructed. ASN.1 data types used are INTEGER, OCTET STRING, OBJECT IDENTIFIER, NULL, SEQUENCE, and SEQUENCE OF. The data types not used in SNMP are BOOLEAN, OBJECT DESCRIPTOR, EXTERNAL, REAL, ENUMERATED, SET, and SET OF. Object types also must be encoded to be sent across a network. To keep the network management process simple, SNMP uses only a subset of the

BER of ITU-T for transfer syntax. Definite length form and nonconstructor encodings are used (see Chapter 5 for a discussion of these terms).

SMI permits two kinds of constructor types:

- *list* has the following syntax:

```
list ::= SEQUENCE { <type1>,      <typeN>}
```

"type1" stands for one of the ASN.1 primitive types defined in SMI. DEFAULT and OPTIONAL clauses are not allowed within the SEQUENCE of a list. We can imagine "list" as being analogous to a one-dimensional row or a column.

- *table* is defined as

```
table ::= sequence of <entry>
```

Here "entry" corresponds to the list constructor. "table" is two-dimensional and has one or more rows, with each row having one or more columns.

RFC 1155 states how the objects are to be defined. The syntax of an object is as follows:

- *Object name:* Should consist of the textual name along with the OBJECT IDENTIFIER. The textual name is known as OBJECT DESCRIPTOR. An example is sysLocation {system 6}. Here, sysLocation is the OBJECT DESCRIPTOR, and {system 6} is the OBJECT IDENTIFIER. In the Internet, the conventions for naming objects are similar to those used in ITU-T; however, the object name must be unique and easily identifiable, and it must be a printable string. {system 6} states that sysLocation is a child under "system" and is the sixth node in the object registration hierarchy. Note here that the object registration hierarchy in the Internet is similar to that of ITU-T.
- *Syntax:* Refers to the syntax used by the object type and should map to one of the permitted ASN.1 data types. sysLocation object type syntax is DisplayString (SIZE (0..255)), for example, DURHAM2154.
- *Definition:* Contains an unambiguous description of the object type. This description is useful for instantiating object types.
- *Access:* States how the object can be accessed by management operations and is the minimum level of support for an object type. The values are read-only, read-write, write-only, and not accessible. With read-only we cannot change the value using the SNMPv1 operation SetRequest. The operations that can be performed for read-only are

GetRequest and GetNextRequest. The access right of write-only enables the SetRequest operation to be done on an object. With read-write, GetRequest, GetNextRequest, and SetRequest operations can be performed. We will examine the SNMPv1 operations in detail later in this chapter.

- *Status:* Refers to the implementation support to the object type. The values are mandatory, optional, or obsolete.

SMI standard RFC 1155 also defines the following object types:

- *Network address:* Presents options for stating addresses in different protocols. At present, only the Internet family of network addresses is allowed.
- *IP address:* Represents a 32-bit Internet address and is represented using OCTET STRING. For BER, only primitive encoding is allowed.
- *Counter:* A nonnegative integer value that increases from 0 to the maximum value of 2^{32}–1. After reaching the maximum value, the counter becomes 0 and increases in value. This is also known as *wrapping around.*
- *Gauge:* A nonnegative integer that can increase or decrease in value. The maximum value allowed for a gauge is 2^{32}–1. One difference between a counter and a gauge is that in a gauge no wrapping around is allowed.
- *TimeTicks:* A nonnegative integer that represents elapsed time, in hundredths of a second, since a particular action. The description of the object type must clearly state the particular action, such as the last update of an object.
- *Opaque:* Permits an object to transmit any data as an OCTET STRING. It is similar to the ASN.1 EXTERNAL type. Note here that an NMS and agents must have previous knowledge of the data type to parse the data.

SMI document RFC 1155 covers the object information model and a set of generic types used to describe management information. To extend the ASN.1 grammar in SMI definitions, the OBJECT-TYPE MACRO is included.

Let us take an arbitrary example of an object to illustrate the meanings of the different terms used.

```
tokenRingPrice OBJECT-TYPE
         SYNTAX INTEGER
         ACCESS read-write
         STATUS optional
         ::= {enterprises 100}
```

In the preceding object definition, tokenRingPrice is the OBJECT DESCRIPTOR, and the syntax of this object is INTEGER. {enterprises 100} is the OBJECT IDENTIFIER. ACCESS is read-write, stating that the values can be read or changed. STATUS optional states that this is an optional object.

For more details on SMI, refer to RFC 1155 (Reference 8.1).

8.4 Management Information Base (MIB-II)

In Internet, as with the ITU-T, the details of objects are stored in a database called the management information base (MIB). MIB is an abstract concept applied to storing data. In the MIB, object types are described by the OBJECT DESCRIPTOR along with OBJECT IDENTIFIER. MIBs are defined by different RFCs. The earlier MIB versions followed RFC 1065, Structure and Identification of Management Information for TCP/IP-based Internets, and RFC 1066, Management Information Base Network Management of TCP/IP-based Internets. RFC 1065 specified how the objects were to be defined and has been superseded by RFC 1155. RFC 1066 defined the objects, but was made obsolete by RFC 1156.

RFC 1155, Structure and Identification of Management Information for TCP/IP-based Internets, specified how the objects were to be defined. This RFC attempted to structure the definition of objects more in the manner of the ITU-T managed object class definitions. RFC 1156, known as MIB-I, contained the definition of objects. This became the Full Standard Protocol. Extensions were added to MIB-I in RFC 1158, and this version is known as MIB-II. This RFC added new objects to different groups and deprecated (i.e., marked for removal from the next versions) some objects. The idea behind these changes was to provide support for multiprotocol entities, as well as readability, clarity, and cleaning up.

RFC 1212, Concise MIB Definitions, provided methods to clean up and remove the redundant object descriptions. RFC 1213, MIB-II for Network Management of TCP/IP-based Internets: MIB II, is yet another improvement over RFCs 1156 and 1158. It adds and refines objects defined in RFCs 1156 and 1158, and also makes use of RFC 1212.

However, readers should consult the latest IAB Official Protocol Standards, published often, for information about the current standardization maturity and requirement levels. Changes are frequently being made to

accommodate new requirements and compatibility with OSI/ITU-T systems management.

8.4.1 Internet Registration Hierarchy

Objects must be uniquely identified for manipulation for management purposes: This is accomplished by using OBJECT IDENTIFIERs. OBJECT IDENTIFIERs are a series of identifiers derived by tagging the numbers attached to the nodes from the root in the registration hierarchy. These numbers are separated by periods.

In the registration tree shown in Chapter 4, Figure 4.7, the registration tree at the starting point has three children with labels of itu-t(0), iso(1), and joint-iso-iut-t(2). Here, itu-t stands for the textual description of the International Telegraph and Telephone Consultative Committee, and 0 stands for the position within the root. Similarly, iso(1) stands for the ISO, and joint-iso-itu-t(2) indicates joint administration by ISO and ITU-T.

The administrative control of each of the subtrees for assigning the numbers below them is the responsibility of the nodes. However, this may be delegated as we go down the tree. Under iso(1), org(3) is assigned to national standards organizations. The two nodes below org(3) have been assigned to the U.S. National Institute of Standards and Technology (NIST). Of the two, one node, dod(6), has been assigned to the Department of Defense (DOD), U.S. Here, internet(1) is a child under dod(6).

There are four children under internet(1), as shown in Figure 8-5: directory(1), mgmt(2), experimental(3), and private(4). The child directory(1) is reserved for the future use of OSI directory services. The Internet MIB-II is defined as a first child under mgmt(2). The IAB has delegated the authority to assign numbers in this subtree to the Internet Assigned Numbers Authority. Thus, the object identifier would be 1.3.6.1.2.1. Here, 1 is from iso; 3 is from org; 6 is from dod; 1 is from internet; 2 is from mgmt; and the final 1 is from MIB-II.

The experimental(3) subtree is meant for objects used in Internet experiments. As an example, if an experimenter gets a number, 12, then the object identifier is 1.3.6.1.3.12. Here, except for the last two numbers, the registration numbers are the same. The next-to-last number, 3, is from the experimental node, and 12 is the number that has been assigned to the experimenter by the Internet Assigned Numbers Authority.

However, the private(4) subtree is used for numbering the objects registered by private bodies. For example, the ABCD Company may ask for a number for registering objects. If it receives a number of 100 from the

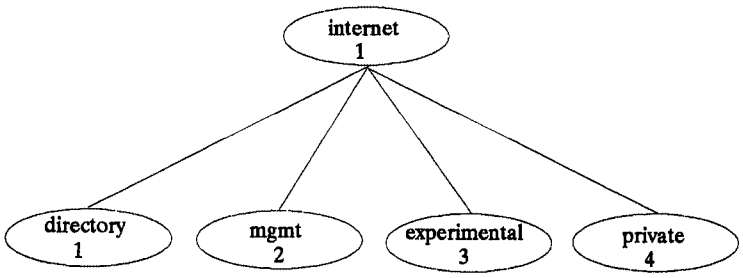


Figure 8-5
Internet registration hierarchy.

Internet Assigned Numbers Authority, it will be under the private(4) subtree. The ABCD Company can define its own objects under this registration number of 100; for example, it may define an adapter for token ring as 25. Then the object identifier for token ring is 1.3.6.1.4.–.100.25. Here, the first four numbers are as explained earlier. The registration number of 4 indicates private, and 100 is the number for the ABCD Company. As already mentioned, the Internet Assigned Numbers Authority assigns these private numbers.

8.4.2 Object Instance Identification

To know the value of an instance of an object, it is necessary to identify the instance. This is done using the OBJECT IDENTIFIER. The following conventions apply for identifying object instances:

- *Scalar objects:* These have only one instance associated with an object. An example of a scalar object is snmpInBadValues. For snmp group, the OBJECT IDENTIFIER for the snmp subtree is 1.3.6.1.2.1.11. By extending the snmp subtree, the snmpInBadValues OBJECT IDENTIFIER is 1.3.6.1.2.1.11.10. We have concatenated the subidentifier of 10 to the snmp subtree to get the OBJECT IDENTIFIER of snmpInBadValues. For each scalar object of snmpInBadValues, there is only one instance, and this instance is identified again by concatenating a value of 0 to the OBJECT IDENTIFIER of snmpInBadValues.
  So an instance identifier for the instance of snmpInBadValues is 1.3.6.1.2.1.11.10.0. The same procedure for deriving an instance identifier is shown as follows:

```
                                    OBJECT IDENTIFIER
snmp subtree                        1.3.6.1.2.1.11
snmpInBadValues                     1.3.6.1.2.1.11.10
An instance of snmpInBadValues      1.3.6.1.2.1.11.10.0
```

- *Columnar objects:* Only objects in a table, known as columnar objects (Figure 8-6), can be manipulated by SNMP. Tables in SNMP are two-dimensional. Instances of objects in tables are identified by the INDEX clause or the AUGMENTS clause. The INDEX clause refers to a row in a table. Note that the AUGMENTS clause is an addition in SNMPv2. Let us take the example of ipAdEntIfIndex in ipAddrTable and see how an instance identifier is formed. The OBJECT IDENTIFIER of the ip subtree is 1.3.6.1.2.1.4. The OBJECT IDENTIFIER of ipAddrEntry is obtained by concatenating subidentifiers 20.1 to the ip OBJECT IDENTIFIER. Hence, the OBJECT IDENTIFIER of ipAddrEntry is 1.3.6.1.2.1.4.20.1. ipAddrTable has a value of 20 in the IP group and ipAddrEntry has a value of 1 in ipAddrTable. The INDEX clause for ipAddrEntry has ipAdEntIfIndex (RFC 1213). The OBJECT IDENTIFIER of ipAdEntIfIndex is derived by concatenating the column number of 2 to the OBJECT IDENTIFIER of ipAddrEntry, which is 1.3.6.1.2.1.4.20.1.2. Now any row of ipAdEntIfIndex is indexed by the INDEX clause subidentifiers associated with ipAdEntIfIndex.

	OBJECT IDENTIFIER
ip subtree	1.3.6.1.2.1.4
ipAdEntIfIndex	1.3.6.1.2.1.4.20.1.2
Fifth instance of ipAdEntIfIndex	1.3.6.1.2.1.4.10.1.2.(fifth row value of ipAdEntIfIndex)

- *Conceptual rows and tables:* When the object types are just rows and tables, they are known as *conceptual rows* and *conceptual tables*. There are no instance identifiers associated with the conceptual rows and tables, and they are not accessible by SNMP operations.
- *Lexicographic ordering:* *Webster's New Collegiate Dictionary* defines *lexical* as "of or relating to words or the vocabulary of a language as distinguished from its grammar or construction." OBJECT IDENTIFIERs are arranged in increasing value in SNMP MIBs. Referring to the previous example of the ip subtree, the OBJECT IDENTIFIER of ipAdEntIfIndex is lexicographically greater than the OBJECT IDENTIFIER of ip. This lexicographic ordering is useful for traversing a table without a management station actually knowing the OBJECT IDENTIFIER of an entry. For example, to retrieve the fifth instance of ipAdEntIfIndex, four GetNextRequest PDUs will be enough to obtain the value of the fifth instance.

The following rules are used for instance identification using the INDEX clause:

- *Integer-valued:* Has a single-value subidentifier to identify an object instance. These are nonnegative integers. The value of ifIndex, which

can take any value ranging from 1 to the value of ifNumber, is an example of an integer-valued INDEX clause.

- *String-valued (fixed-length strings):* Has *n* octet length with *n* subidentifiers.
- *String-valued (variable-length strings):* Has *n* + 1 octet length with *n* subidentifiers. The first octet has the value of *n*, which indicates the number of subidentifiers.
- *Object-identifier-valued:* Used with object identifiers. The first value is the number *n* of subidentifiers used along with the *n* subidentifiers.
- *IpAddress-valued:* Associated with four subidentifiers familiar in the IP address of the type a.b.c.d.
- *NsapAddress-valued:* Uses syntax similar to a fixed-length string with the length of the string as *n*.

8.4.3 Table Manipulation

In MIB-II tables (Figure 8-6), each object is represented by a column; the value of each object instance is represented in a row. The way to retrieve values from a table is somewhat nonintuitive. The length of each column is traversed in its entirety, and then one moves on to the next column. To add the value of an object instance, the value is entered in a row with a SetRequest operation. To delete an entry, again using the SetRequest operation, the value is set to invalid.

Whether or not to remove the invalid entries is left as an implementation option. It is better to remove invalid entries when storage is a constraint; traversing the table will be faster with fewer entries. But this involves extra work—now and then, some sort of cleanup of the table to remove invalid entries must be done. If it is decided to do this cleaning,

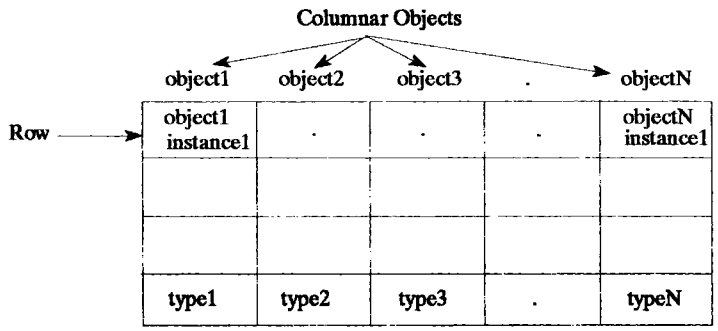


Figure 8-6
Concept of tables.

then it must also be determined how frequently to remove the invalid entries. Ultimately, these decisions involve trade-offs that should be examined during the design stage.

8.4.4 MIB-II Details (RFC1213)

It is necessary to follow rules when defining new MIBs. Old object types are not deleted, but they may be deprecated. The semantics of old object types should not be changed between versions: if it is necessary to change the semantics, new object types must be formed. Some of these rules are formulated because of the coexistence of multiple versions of MIB.

In MIB, only the essential objects are defined. This has been done to make the implementations simple. However, there is enough flexibility to define the implementation of specific objects. The guidelines for defining new objects are provided by SMI. New objects can be added under the experimental subtree or under the enterprises subtree. Also, new versions of MIB can be released for the new standard objects, by which the definition of objects is sufficiently flexible in Internet.

MIB-II has added the data type, DisplayString. DisplayString is defined as an OCTET STRING, and is a printable ASCII character string ranging in size from 0 to 255 octets. Another data type of PhyAddress is again an OCTET STRING used to represent media addresses. As an example, a token ring address can be represented in binary in six octets.

Objects in the Internet are classified into different groups. These groups are under {mgmt 2}, as shown in Figure 8-7, and the classification into groups assists in easily assigning object identifiers to the groups. Also, the objects under these groups must be implemented as a group. For example, if the TCP group is implemented, then all objects under the TCP group, such as tcpRtoAlgorithm and tcpRtoMin, must be implemented. For more details, MIB-II objects, refer to RFC 1213 (Reference 8.4).

8.5 SNMPv1 Protocol and Protocol Details

SNMPv1 PDUs are furnished in Table 8-1. Note that these PDUs use a similar data format. When one protocol entity sends a GetRequest, GetNextRequest, or SetRequest PDU, the response is a GetResponse PDU

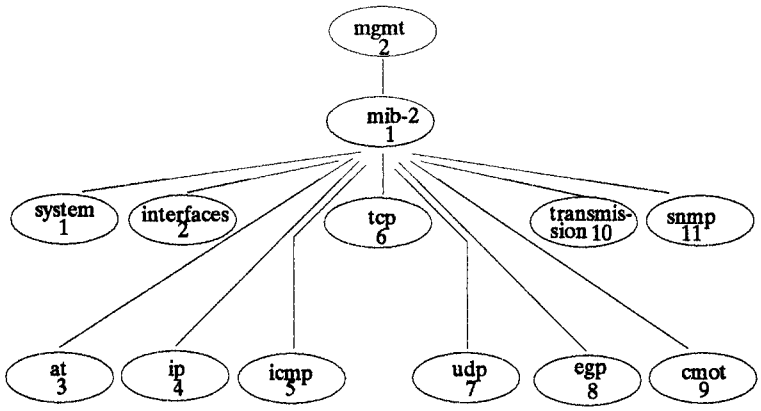


Figure 8-7
MIB-II objects group.

(Figure 8-8). When there are errors or special cases, traps are sent from one protocol entity to another one. An SNMPv1 trap is used for reporting an asynchronous occurrence of an event such as a serious error, important state change, or crossing of a threshold. It is usually reported from an agent to a manager, and is not confirmed. Event types are defined in TRAP-TYPE. In SNMP especially, GetNextRequest assumes that object values in the MIB are arranged in a tabular form.

The PDUs GetRequest, GetNextRequest, SetRequest, and GetResponse have request-id, error-status, error-index, and variable-bindings. Values for error status and error index for GetRequest, GetNextRequest, and SetRequest PDUs are always taken as 0. Request-id is used as an identifier for correlating the responses or for identifying duplicate responses. error-status indicates the kind of error. These are listed in Table 8-2. error-index gives the position of the variable which was responsible for the error. Variable refers to an instance of an object, and variable-binding refers to the combination of a variable name and value. There can also be a list of variable names and values.

TABLE 8-1

SNMPv1 Protocol Data Units.

GetRequest
GetNextRequest
GetResponse
SetRequest
Trap

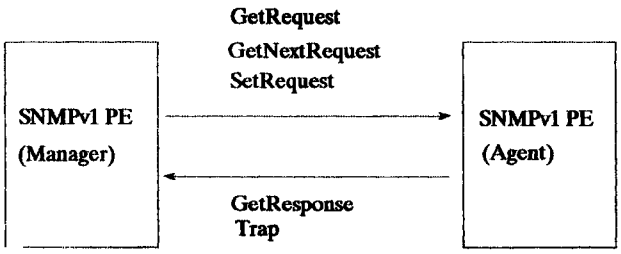


Figure 8-8
SNMPv1 management protocol PDUs.

SNMP uses a combination of traps and polling for retrieving management information. Traps are expected to be sparingly used. When there is an error, a trap is sent from an agent to the NMS. Once this trap is received, the NMS must send a response. Further management information is retrieved using polling.

8.5.1 Functioning of SNMPv1 PDUs

As mentioned earlier, SNMP is used for carrying messages from one *protocol entity* (PE) to another. A PDU is constructed in ASN.1 form and is then sent to an authenticator, along with the community name and source and transport addresses. The authenticator checks whether the sending and receiving protocol entities can really exchange messages. The result of the check, which can be an authentication failure or an ASN.1 object, is sent back to the PE. If the message has passed the authenticator check, it is formatted using BER and sent by the transport mechanism to the receiving PE. Generating an SNMPv1 message is shown in Figure 8-9.

On the receiving side, the PE parses the data received and checks for version numbers. If the PDU fails during this action, then the datagram is

TABLE 8-2
SNMPv1 Protocol Errors.

noError
tooBig
noSuchName
badValue
readOnly
genErr

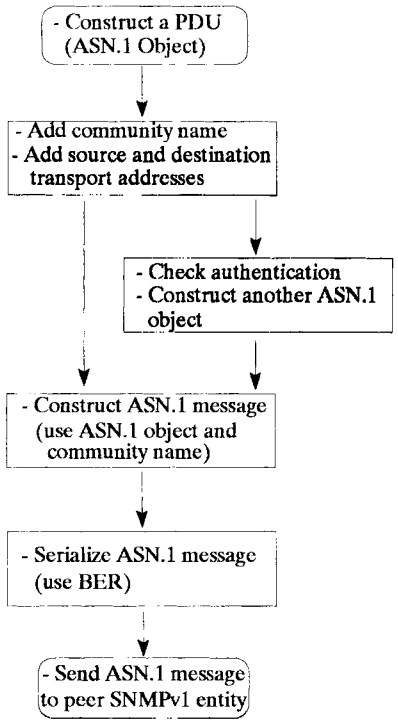


Figure 8-9
Generating an SNMPv1 message.

discarded. After this operation, user data, community name, and source and destination addresses go to the authenticator. The authenticator checks whether the receiving PE can actually receive data from the sending PE. The result can be authentication failure. In such a case, a trap may be generated and sent to the sending PE. However, sometimes this trap is just logged and a record is maintained for any remedial action to be taken. On the other hand, if the authenticator passes the authentication process, the receiving PE processes the message, takes the appropriate action, and sends the response back to the sending PE. The steps involved in the receiving end are shown in Figure 8-10.

Here we must note that some management applications may not need the security provided by authentication. In such implementations, the authentication steps may be skipped altogether for management applications. Also, in some cases, additional security steps such as extra access control functions, may be needed. Thus, the level of authentication required is guided by implementation considerations.

When SNMPv1 protocols are used, errors such as those listed in Table 8-2 can be generated. The value of noError is 0; from there, values pro-

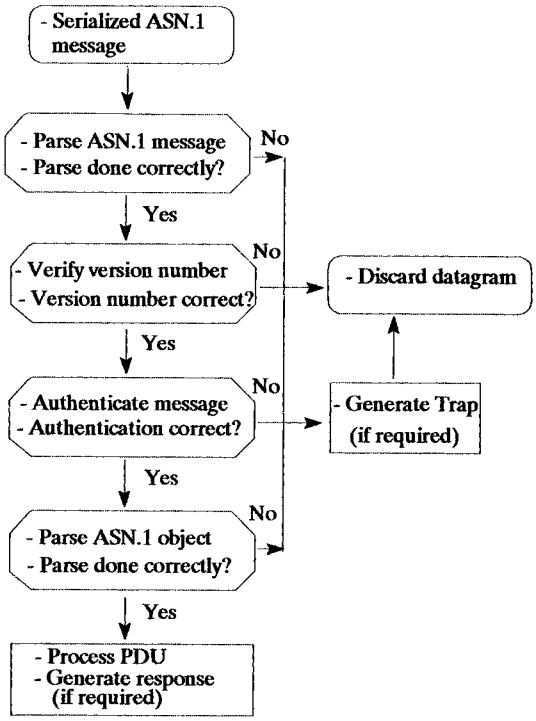


Figure 8-10
Receiving an SNMPv1 message.

gressively increase by 1. Thus, badValue has a value of 3. Let us examine the individual SNMPv1 PDU function. When a GetRequest PDU is sent, the receiving PE checks whether the object name is correct. If the object name is not correct, then the error status is noSuchName. Afterward, a check is made as to whether the object type is aggregate. Because SNMP does not allow aggregate object types, the error status of noSuchName is set for the aggregate object type cases.

A response for GetRequest must be within the capacity of the local PE. If it is not possible to generate responses, then the error status is set to tooBig. When processing a GetRequest, there may be errors. In such a case, the error status is set to genError. When there are no errors, the error status is set to noError. Responses in all cases are sent in the form of a GetResponse PDU.

GetNextRequest is similar to GetRequest except that we retrieve the next value instead of the one furnished, and the PDU type is different. Here, the assumption is that MIB has object names arranged in a lexicographical order. However, if a lexicographical successor is not available, then error status is set to genErr.

In SetRequest, too, if a variable name is not available, then the error status is set to noSuchName. If the value of a variable whose value must be changed to that furnished in SetRequest is not in conformity with ASN.1 language, type, length, or variable, the error status is set to badValue. The case of error status set to tooBig is similar to that explained earlier. If the value of a variable cannot be changed, it is classified under genError. However, if the value of a variable is changed to the value mentioned in SetRequest, then the reply PDU of GetResponse has its error status set to noErr.

A trap is used to present problems or changes in the Internet. A trap PDU consists of the object type generating the trap, the address of the object generating the trap, the generic trap type, the time stamp when the trap was generated, and any variable binding carrying relevant information. If the enterpriseSpecific trap is present, the specific trap field is present. Generic traps are listed in Table 8-3. In this table, just as in Table 8-2, coldStart has a value of 0 and the values for other traps increase by 1. Hence, linkUp has a value of 3 associated with it.

Let us briefly review the meanings of different generic traps. The first trap is coldStart, which implies that an agent is reinitializing and the configuration and implementation details might have changed. Thus, the station receiving this trap will have to reset itself, whereas in the case of warmStart, there are no changes in the configuration and implementation details when reinitializing.

The traps linkDown and linkUp are used when the communication links used are affected, and the affected link name and value are furnished in variable bindings. When a protocol message fails authentication, then the authenticationFailure trap is generated. egpNeighborLoss is generated when an EGP neighbor is no longer available. When traps can-

TABLE 8-3

SNMPv1 Traps Generated.

coldStart
warmStart
linkDown
linkUp
authenticationFailure
egpNeighborLoss
enterpriseSpecific

not be specifically classified into any one of these categories, the enterpriseSpecific trap comes in handy. This trap can also be used to convey any specific trap that may be implementation-specific. This is carried in the specific trap field.

8.6 Proxy

The *proxy* concept is very popular. It is possible that a management station may not be able to manage a device such as a modem, bridge, or router because the device responds to proprietary commands. In such a case, a proxy agent is used, as shown in Figure 8-11. Between a management station and a proxy agent, standard protocols are used. However, between the proxy agent and the managed devices, proprietary protocols are used. The proxy assumes the role of an agent with respect to the manager and acts as a manager for managed devices.

One of the main functions of a proxy agent is protocol conversion. As an example, the standard command from the management station is converted or mapped to the semantically equivalent proprietary command and sent to the managed devices. As mentioned earlier, managed devices understand these proprietary commands. Similarly, responses to the commands sent to managed devices are mapped back to standard responses in the proxy agent.

The conversion from standard protocols to proprietary protocols can be achieved by two methods. One is the mapping or translation of each field of standard protocols to each field of proprietary protocols. However, in some cases, one-to-one mapping of fields may not be possible,

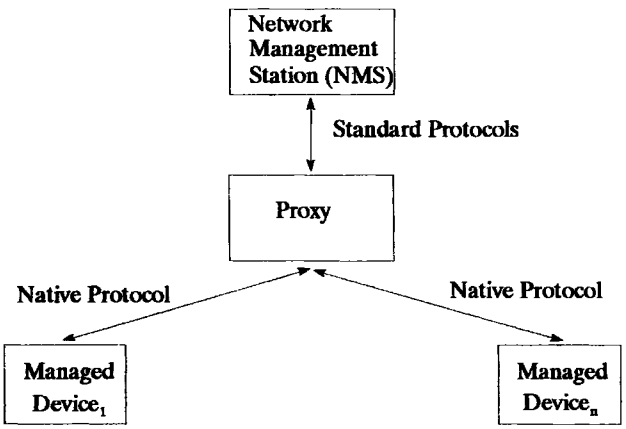


Figure 8-11
The proxy concept.

because there may not be similar, functionally equivalent fields. In such cases, the best possible assumptions will have to be made.

Another method involves encapsulating the standard protocols within the data fields of the proprietary protocols. This information will have to be retrieved by the software or hardware in the managed devices, but the managed devices will have the added responsibility of interpreting standard protocol fields. In some cases, the information carried in standard protocols may prove redundant or not amenable for proper interpretation. In either case, it is possible that some information will be lost due to the inability to use one-to-one mapping or the inability to retrieve all the information. To overcome this, the proprietary protocols are normally continuously modified to keep abreast of the standards.

Furthermore, the introduction of a proxy agent adds complexity. The mapping of standard protocols to the proprietary protocol requires resources. Note that we are adding one more level to the communication between the management station and managed devices. Also, if the proxy is in a separate workstation, there are considerations such as additional backup in case of the failure of the backup. All these add to the complexity.

Also, a proxy agent provides the flexibility to enforce "under-the-cover" access policies. The SNMP MIBs do not have to bother about them. The access policy can be for managed devices represented as objects, and it will be handled between the proxy agent and the managed devices.

A proxy can also be used for selectively forwarding information to a management station. It may log the information it receives from managed devices, and may maintain details of aggregate objects. Aggregate objects are defined for combining the details on more than one object. The management system may ask the proxy to send the details on aggregate objects. The aggregate objects may also be defined for summary statistics, configuration details, and so on.

8.7 SNMP over Different Protocols

SNMP has been defined over different popular protocols. Some of these are:

- SNMP over IPX (RFC 1420)
- SNMP over Ethernet (RFC 1089)
- SNMP over AppleTalk (RFC 1419)
- SNMP over OSI (RFC 1418)

8.8 SNMPv2

The original SNMP did not have elaborate security arrangements. To overcome this major weakness, SNMP version 2 (SNMPv2) was released. Accordingly, the original SNMP is known as SNMP version 1 or SNMPv1.

After SNMPv2 was released, there was much controversy on the administrative and security frameworks. One of the primary complaints on administrative and security frameworks was that they were complex, and it was difficult for the network administrators to use the administrative and security frameworks. As a result, SNMPv2 work got bogged down in controversies for some time.

The SNMPv2 security model was based on the concept of *parties*. Because of the disagreements on SNMPv2 security, two different approaches to security were taken in SNMPv2u and SNMPv2*. SNMPv2u advocated minimum standardization and was primarily focused on providing solutions to small agents: Security issues involving large networks were left open to be solved later.

In contrast, the SNMPv2* approach suggested standardization of security such that large networks were also covered. SNMPv2* also addressed remote configuration of security as well as scalability issues. SNMP version 3 or SNMPv3 started from these two approaches. We will look into SNMPv3 later. Those interested in the discussion on some of the controversies on SNMPv2 should consult Reference 8.21.

It was felt that the good work done in SNMPv2 should be reused. So, the least controversial RFCs in the SNMPv2 documents were accepted as draft standards. SNMPv2 documents from RFC 1902 through RFC 1908 (Table 8-4) are draft standards. RFC 1901, Introduction to Community-based SNMPv2; RFC 1909, An Administrative Infrastructure for SNMPv2; and RFC 1910, User-based Security Model for SNMPv2, were changed to the status of experimental. RFC 2089, Mapping SNMPv2 onto SNMPv1 within a Bilingual SNMP Agent, was given the status of informational. All these RFCs were part of the original SNMPv2.

We will discuss only the salient points of SNMv2 and the RFCs in the draft standard status. Bearing this point in mind, SNMPv2 has the following enhancements over SNMPv1:

- There have been major enhancements to the structure of management information of SNMPv1. MIB-II has been extended to include managed objects for SNMPv2. The syntax and semantics of managed objects have been refined. Also textual conventions have been

TABLE 8-4

SNMPv2 Draft Standard Documents.

RFC Number	Title of RFC
1902	Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)
1903	Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)
1904	Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)
1905	Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)
1906	Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)
1907	Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)
1908	Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework

enhanced to extend the semantics of the standard object types defined in SMI.

- More operational PDUs are added. GetBulkRequest has been added to retrieve a large amount of management information. InformRequest has been added for communication between managers in a distributed environment. Though InformRequest was proposed for communication between managers, in SNMPv2 and SNMPv3 it is popular for sending notifications.
- Conformance statements have been added and a strategy for the migration of SNMPv1 to SNMPv2 has been provided.
- The transport used for carrying SNMPv2 protocols has been made more flexible and defined for different environments.

8.8.1 SNMPv2 Structure of Management Information (SMI)

The Structure of management information for SNMPv2 RFC 1902 (Reference 8.5) describes rules for defining modules, objects, and traps. Note that module is a collection of related objects. SMI provides the following definitions:

- *Module definitions:* The ASN.1 macro MODULE-IDENTITY is used to provide the syntax and semantics when defining a module.
- *Object definitions:* The ASN.1 macro OBJECT-TYPE provides the syntax and semantics to define an object.
- *Notification definitions:* The ASN.1 macro NOTIFICATION-TYPE is used to describe a *notification.* A notification is an unsolicited confirmed message of an event within an InformRequest-PDU.

8.8.1.1 Information Module. An *information module* is an ASN.1 module and provides information for network management. An information module is identified by a MODULE-IDENTITY macro. It provides the contact and revision history of the information modules. There are three types of information modules:

- *MIB modules:* Contain information on related objects.
- *Compliance statements:* Provide compliance requirements for MIB modules.
- *Capability statements:* Contain information on the capabilities of the agent implementation.

An information module need not have all the three modules. Compliance and capability statements will be discussed in Section 8.8.2.

8.8.1.2 Explanation of New Terms. Many new terms have been added in RFC 1902 (Reference 8.5) over the SNMPv1 SMI. Some of the new terms that need explanation are:

- *BITS:* Used for enumeration of named bits; starts from a value of 0.
- *Unsigned32:* Represents an integer value from 0 to $2^{32} - 1$.
- *Counter64:* A wraparound counter with a maximum value of $2^{64} - 1$.
- *UNITS:* Has textual definition of the units of measurements, such as hours and minutes, associated with an object.
- *MAX-ACCESS:* Relates to the maximum level of access allowed for an object. Note the distinction between the access rights of read-write and read-create. read-create is the maximum level of access allowed. With the read-create access right, an object instance can be created in addition to the read-write permission.
- *INDEX:* Used for object instance identification in a conceptual row. A new keyword, IMPLIED, is used for objects with variable-length syntax.

- *AUGMENTS:* An alternative to an INDEX clause, it extends the conceptual row with reference to a base conceptual row provided by an INDEX clause. AUGMENTS is useful for extending the definitions of tables. As an example, if Table 2 is a logical extension of Table 1, then the AUGMENTS clause may be used to connect these two tables.

The use of the Opaque data type (Section 8.3) is retained in RFC 1902 for backward compatibility reasons and should not be used for defining new object types. Reference 8.20 explains well the changes made to RFC 1902, RFC 1903, and RFC 1904.

8.8.1.3 Naming of Objects. The naming hierarchy is shown in Figure 8-12. The snmpV2 object subtree is contained in the internet subtree. We have already discussed mgmt(2), experimental(3), and private(4) in Section 8.4.1.

Under the snmpV2 subtree are snmpDomains(1), snmpProxys(2), and snmpModules(3). snmpDomains is used for transport mapping, snmpProxys is for transport proxies, and snmpModules is for the module identities.

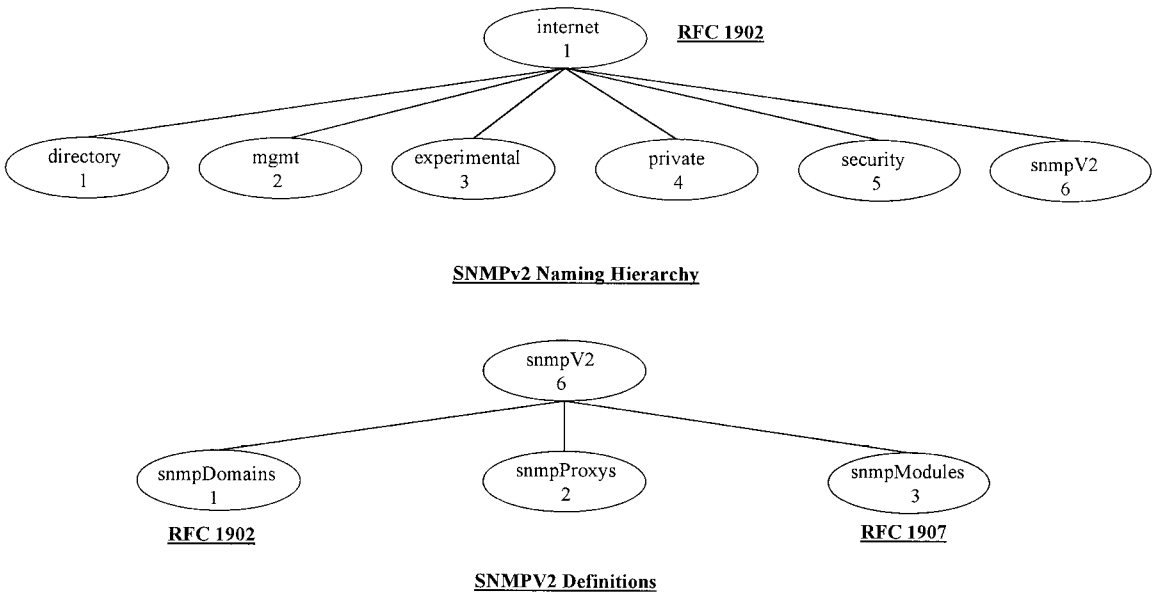


Figure 8-12
SNMPv2 naming hierarchy and object definitions.

8.8.1.4 SNMPv2 Textual Convention. The SNMPv2 textual convention is used for extending the semantics of standard data types in SMI. With this provision, SMI need not be modified often. RFC 1903 (Reference 8.6) defines a base set of textual conventions that use the conventions of the TEXTUAL-CONVENTION ASN.1 macro.

- *DisplayString (OCTET STRING (SIZE(0..255)))*: Represents the textual convention using the ASCII character set. It is used for easy readability. The length of DisplayString should not exceed 255 octets.
- *PhysAddress (OCTET STRING)*: Used to indicate the media address.
- *MacAddress (OCTET STRING (SIZE(6)))*: Represents the MAC address as defined by IEEE 802.1 with the least significant bit transferred first.
- TruthValue *(INTEGER)*: Provides a means to indicate a Boolean value, with true being 1 and false being 2.
- *TestAndIncr (INTEGER (0..2147483647))*: Tests first whether a value provided using a management protocol matches the value of the object instance. If the value matches, then the value of the object instance is increased by 1. If the value of an object instance reaches the maximum value of 2147483647, then the object instance value wraps to 0. However, a mismatch between the value provided and the value of the object instance results in an *inconsistent value* error.
- *AutonomousType (OBJECT IDENTIFIER)*: Can be used to extend objects with a distinct identifier. Here the key is "autonomous," and the extension must be easily distinguishable. The textual convention can be useful for defining additions to the hardware and protocols.
- *VariablePointer (OBJECT IDENTIFIER)*: Is a pointer to a specific object instance.
- *RowPointer (OBJECT IDENTIFIER)*: Is a pointer to a conceptual row.
- *RowStatus (INTEGER)*: Used to manipulate the creation and deletion of managed objects in conceptual rows. RowStatus has the values of active, notInService, notReady, createAndGo, createAndWait, and destroy.
- *TimeStamp (TimeTicks)*: The same as sysUpTime, which is the value in TimeTicks when an agent was last reinitialized.
- *TimeInterval (INTEGER (0..2147483647))*: Represents the interval in units of 0.01s between two time periods.
- *DateAndTime (OCTET STRING (SIZE 8 | 11))*: Used to indicate the date and time. The length of DateAndTime can be 8 or 11 octets. The different fields have significance, as shown in Table 8-5. We have already

TABLE 8-5

DateAndTime Representation.

Field	Octets	Contents	Range
1	1—2	Year	0..65536
2	3	Month	1..12
3	4	Day	1..31
4	5	Hour	0..23
5	6	Minutes	0..59
6	7	Seconds (use 60 for leap second)	0..60
7	8	Deciseconds	0..9
8	9	Direction from UTC	"+"/"–"
9	10	Hours from UTC	0..11
10	11	Minutes from UTC	0..59

looked into what UTC means in Chapter 5. As can be observed from Table 8-5, we do not need octets 9, 10, and 11 for local time.

8.8.2 Conformance Statements

Conformance statements define the acceptable level of implementation. An implementation can compare the actual functions provided with the benchmark provided in conformance statements to check whether the implementation satisfies the requirements. There are two types of notations used in SNMPv2; these are defined in RFC 1904 (Reference 8.7). The conformance notations are:

- *Compliance statements:* Refer to the minimum set of requirements imposed on one or more MIB modules. This compliance is checked with the ASN.1 MODULE-COMPLIANCE macro.
- *Capability statements:* Refer to the actual functions provided by one or more MIB groups supported by an agent and the ASN.1 AGENT-CAPABILITIES macro. When an agent supports a MIB group, it has to support all the objects in a MIB group.

The objects used for conformance are combined to form a group, and the ASN.1 macro OBJECT-GROUP is used to provide the syntax and semantics for this group. For conformance purposes, RFC 1904 defines

NOTIFICATION-GROUP macro to represent a collection of notifications and AGENT-CAPABILITIES statements to refer to one or more MIB groups.

8.8.3 SNMPv2 Protocol Messages

SNMPv2 uses slightly different protocol messages (Figure 8-13) for communicating management information than does SNMPv1. SNMPv2 protocol messages are explained in RFC 1905 (Reference 8.8). The format of SNMPv2 PDUs such as GetRequest, GetNextRequest, SetRequest, SNMPv2-Trap, Response, InformRequest, and Report are similar to those of SNMPv1 PDUs; however, there are some minor differences, as follows:

- The GetBulkRequest PDU has been added as shown in Figure 8-13. InformRequest and Report PDU are also new PDUs. However, Report PDU is not defined in SNMPv2.
- A trap PDU of SNMPv1 is no longer there, and a PDU type of 4 used for the SNMPv1 trap is obsolete. The new value of the PDU type SNMPv2 is 7 for SNMPv2-Trap.
- The GetResponse of SNMPv1 is renamed the Response PDU in SNMPv2.

A brief explanation of SNMPv2 PDUs is as follows:

- *GetRequest:* When an application requests an SNMPv2 entity, this PDU is generated by the entity and sent to an agent. This message is similar to the SNMPv1 GetRequest message. On the receiving end, this message is analyzed to check whether the request is a proper one. If it is proper, then the error status field is set to noError and the error index field is set to 0 in the response and the response is sent to the manager.
- *GetNetRequest:* This message is similar to the getNextRequest in SNMPv1 messages and is based on the tabular concept of the SNMP MIB. If there are no errors, then the response contains the next

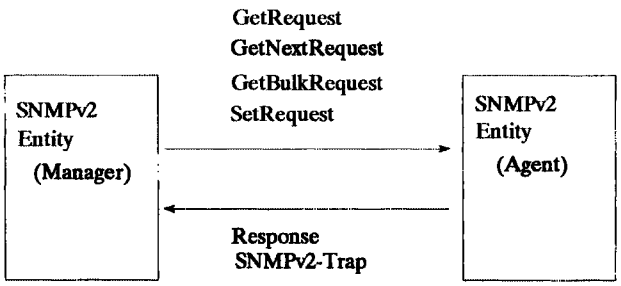


Figure 8-13 SNMPv2 protocol operations.

instance of a variable that has been sent in the request. If there are any errors, then the error status and error index fields correspond to the errors observed.

- *SetRequest:* There are two steps involved. The first is a validation phase; if this succeeds, changes are made. During the validation step, a check is performed of whether change on a variable can be made. If this is not possible, then the error status and error index fields are set accordingly in the response PDU. However, if the validation step is successful, then the value of the variable is set to the value mentioned in the SetRequest PDU. If the variable is not there, it is created and its value is set. In this case, in the response PDU, the error status field shows noError, and the error index field is zero.
- *GetBulkRequest:* This is primarily used for retrieving a large amount of table data from the MIB, unlike the GetRequest and GetNextRequest PDUs. Here also, if there are errors noticed in the receiving end, the error status and error index fields are set. If no errors are noticed, data is sent in the response PDU.
- *InformRequest:* This is a protocol message for conveying MIB information from one SNMPv2 entity to another, both acting in the manager role. This PDU was originally meant for manager-to-manager communication. The receiving SNMPv2 checks to see if there are any errors. If errors are noticed during processing of this PDU, the response PDU has the corresponding error status and error index fields set. If there are no errors, the MIB information is passed on to the application using the receiving SNMPv2 entity and generates a response to the sending SNMPv2. Also note that RFC 1902 on SMI states that the NOTIFICATION-TYPE macro can be contained in an InformRequest PDU. In SNMPv3, InformRequest is used for carrying the notifications.
- *Response:* This PDU is generated as a response for GetRequest, GetNextRequest, SetRequest, GetBulkRequest or InformRequest PDUs sent from a sender SNMPv2 entity. The sender SNMPv2 entity is the manager. The receiving SNMPv2 entity is the agent, and it prepares the response PDU and sends it back to the sending SNMPv2 entity. The sender SNMPv2 must be able to handle the errors generated and pass on the response PDU to the application using the sender SNMPv2 entity.
- *SNMPv2-Trap:* When an exceptional situation occurs in an SNMPv2 entity acting in an agent role, then a trap is generated and sent to the SNMPv2 entity. The trap numbers in SNMPv2-Trap indicate the reasons why a trap has occurred. SNMPv2-Trap is unconfirmed in the

sense that no response is generated from the manager on receiving an SNMPv2-Trap.

- *Report:* This PDU is not defined in RFC 1905. It is primarily used to account future SNMP administrative framework requirements such as to make use of faster error recovery. Currently, it is left to the implementers to define the usage and semantics. An example of the use of a Report PDU is to report time stamp information to a manager for time synchronization.

SNMPv2 and SNMPv3 use trap and InformRequest to report events. InformRequest is confirmed. An agent or a notification originator sends a notification to a manager or notification receiver and waits for confirmation from the manager or notification receiver. If the response is not received from the manager or notification receiver within a configured time, then agent transmits the notification again. These retransmissions will continue until the configured number of retries is reached or a response is received.

Event types are defined in NOTIFICATION-TYPE in RFC 1902 and the status of the event types is furnished by the STATUS clause. STATUS clause is not available in traps.

Reference 8.18 has a very good discussion on trap, event report, and notifications.

8.8.4 Transport Mapping for SNMPv2

RFC 1906 (Reference 8.9) describes the transport mappings. SNMPv2 management protocols must be associated with a transport service for transferring the management information. However, there are different types transport services that can be used for SNMPv2 (Figure 8-14). The term *snmpDomains* stands for the association of the SNMPv2 management protocols with a transport service. The following types of transport service for SNMPv2 have been defined:

- *snmpUDPDomain:* SNMPv2 over UDP is the preferred manner of transferring management information.
- *snmpCLNSDomain:* SNMPv2 over OSI's connectionless-mode transport service (CLTS) is an alternative. Connectionless transport service functions over the connectionless network service.
- *snmpCONSDomain:* Here, SNMPv2 is transferred using the connection-oriented network service (CONS). In this case, connectionless-mode transport service runs over the connection-oriented network service.

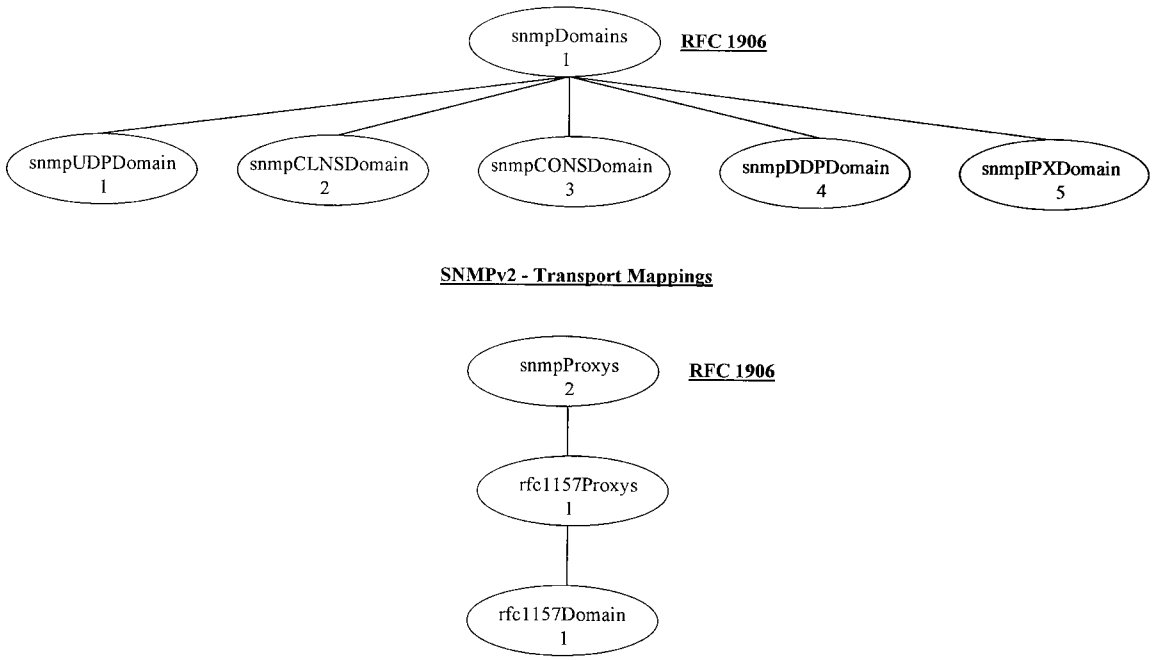


Figure 8-14
SNMPv2 transport mappings and snmpProxys.

- *snmpDDPDomain:* SNMPv2 uses the AppleTalk® DDP for the transport service.
- *snmpIPXDomain:* SNMPv2 uses the NetWare® IPX for the transport service.

8.8.5 SNMPv2 MIB

SNMPv2 MIB is explained in RFC 1907 (Reference 8.10). There are some changes between SNMPv2 and SNMPv1 MIBs. The IMPORT statement refers to MIBs used in SNMPv2, and they are to be used. Under the snmpModules subtree (Figure 8-15), there is the subtree of snmpMIB(1). snmpM2M(2), which was defined in RFC1451, Manager-to-Manager Management Information Base, and partyMIB(3), which was defined in RFC 1447, Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2), have been removed from the SNMPv2 MIB in RFC 1907.

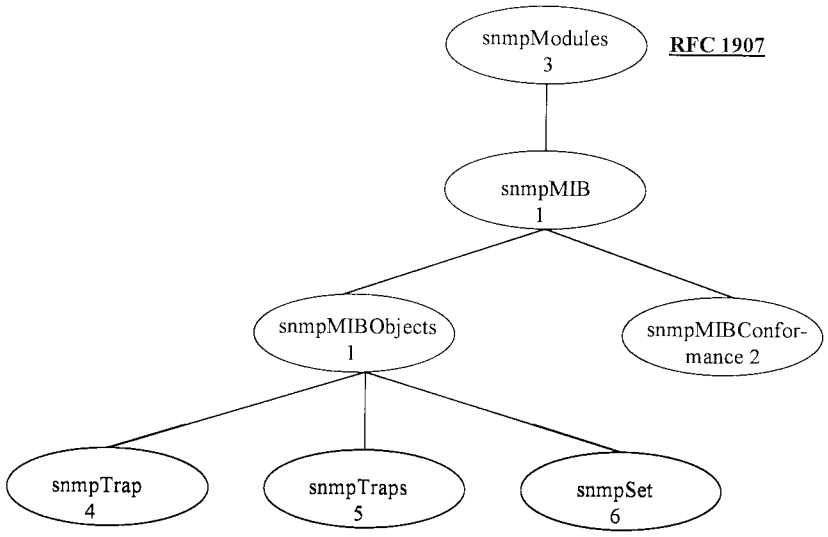


Figure 8-15
snmpModules.

As can be seen from the Figure 8-15, snmpMIB has snmpMIBObjects and snmpMIBConformance subtrees. Again below the snmpMIBObjects, we have snmpTrap(4), snmpTraps(5), and snmpSet(6). All the other object groups under snmpMIBObjects are obsolete.

System group, which was part of MIB-II, has been included in SNMPv2 MIB, in RFC 1907. A new object, sysORUpTime, has been added to sysORTable. sysORUpTime refers to the time stamp when a resource with a particular sysORID value has been instantiated. sysORTable is a conceptual table for an SNMPv2 entity in an agent role.

8.8.5.1 snmpTrap Group. snmpTrap group objects are used by the SNMPv2 entities in the agent role to generate SNMPv2-Trap PDUs. Objects in this group are useful for fault management. In this group, there is one table, snmpTrapTable, which has one entry, snmpTrapNumbers.

8.8.5.2 snmpTraps Group. The snmpTraps group refers to the well-known SNMPv2 traps. We have seen these traps in the SNMPv1; they are useful for fault management. The definitions of linkUp and linkDown traps have been deleted from the snmpTraps Group, as they have become part of RFC 1573, Evolution of the Interfaces Group of MIB-II.

8.8.5.3 snmpSetGroup. The snmpSet group has one object and is used by the SNMPv2 entities acting in a manager role to coordinate the use of

an SNMPv2 set operation. This object can be used for performance management.

8.8.5.4 snmpMIBConformance group. The snmpMIBConformance group provides the compliance statements for SNMPv2 entities implementing SNMPv2 MIB. The compliance statements are provided by the snmpMIBCompliance. snmpMIBgroups refer to the compliance statements for the collection of objects for each of the SNMPv2MIB groups.

8.8.6 Coexistence between SNMPv1 and SNMPv2

For migration from SNMPv1, the proxy approach (see Figure 8-16) has been suggested. In this approach, an SNMPv2 entity may be acting as a manager; the SNMPv1 entity can be in an agent role. However, in this case, the SNMPv2 proxy agent will have to convert the GetBulkRequest PDU to GetNextRequest PDU and set nonrepeaters and maximum repetitions fields to 0. Traps will have to be modified in the proxy agent. In a bilingual approach, a manager functions as a proxy when dealing with an SNMPv2 agent. Here a manager maps SNMPv2 PDUs to SNMPv1 PDUs. Some of the strategies for migration are:

- Upgrading management stations to SNMPv2. If this is done, it is easy to support SNMPv2 agents as they become available.

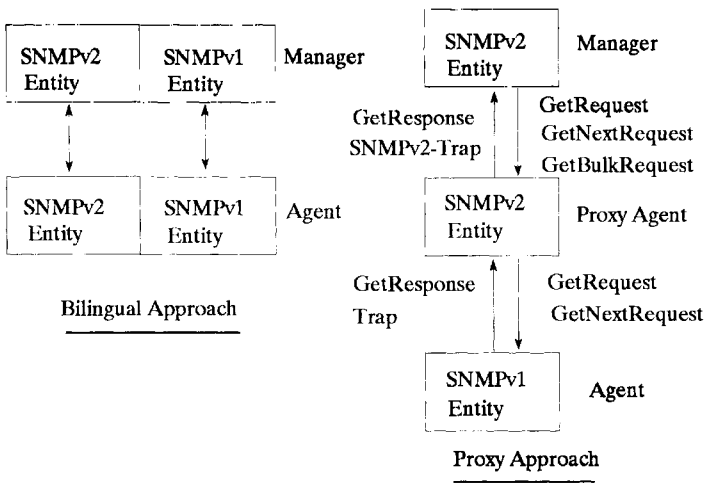


Figure 8-16
SNMPv1 and SNMPv2 coexistence.

- Upgrading and acquiring new SNMPv2 agents, if required. Once the management stations can handle SNMPv2, it is easy to add new SNMPv2 agents.
- Handling the existing SNMPv1 agents; the proxy approach provides easy coexistence.

For more details on the migration and coexistence between SNMPv1 and SNMPv2 refer to RFC 1908 (Reference 8.11).

8.8.7 Device-Dependent Objects

In addition to the objects defined in MIB-II and other SNMPv1, SNMPv2, and SNMPv3 documents, objects are defined for different media types. These are useful, and they can be incorporated in agents.

8.9 SNMPv3

SNMPv3 is an extension of the SNMPv2 framework. It consists of a new SNMP message format, security for messages, and access control. The primary objectives of SNMPv3 were to work on security and administrative frameworks and to reuse, as much as possible, the ideas and work done in SNMPv2u and SNMPv2*. SNMPv3 standards include RFC 2271 through RFC 2275; the list of RFCs is furnished in Table 8-6. These RFCs are proposed standards.

TABLE 8-6 SNMPv3 Standard Documents (Proposed Standards).

RFC Number	Title of RFC
2271	An Architecture for Describing SNMP Management Frameworks
2272	Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)
2273	SNMPv3 Applications
2274	User-based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3)
2275	View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)

Note that SNMPv3 RFCs need clarifications and corrections. As the SNMPv3 standards are in proposed standard status at the time of writing this book, these RFCs are expected to undergo some changes before advancing to other stages of Internet Standard track. Because of the experiences with SNMPv2, acceptance and implementations by the industry will be slow to come by. Therefore, caution is advised in implementing SNMPv3.

8.9.1 SNMPv3 Architecture

SNMPv3 architecture is explained in RFC 2271 (Reference 8.12). The concept of a management system (Figure 8-17) is slightly different from the earlier concepts of SNMPv1 and SNMPv2. A management system definition is based on SNMP entities. A manager in the traditional sense includes a command generator and a notification receiver. And an agent contains a command receiver and a notification generator. It is possible to have more than one agent in the SNMPv3 architecture.

The main concept of SNMPv3 architecture is centered on *SNMP entities*. An SNMP entity is an implementation feature. The components of an SNMP entity are shown in Figure 8-18. An SNMP entity has two main components. It consists of an SNMP engine and one or more applications. There is a one-to-one association between an SNMP engine and the SNMP entity.

An SNMP engine includes a dispatcher, a message processing subsystem, a security subsystem, and an access control subsystem. SNMPv3 applications are part of an SNMP entity. SNMPv3 applications make use of the services provided by an SNMP engine to accomplish specific tasks. Also, SNMPv3 applications may interact with information models and coordinate the processing of management operations. SNMPv3 applications currently include command generators, command responders, notification originators, notification receivers, and proxy forwarders as shown

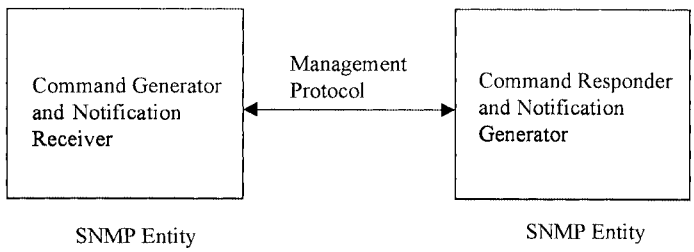


Figure 8-17
SNMPv3 management system.

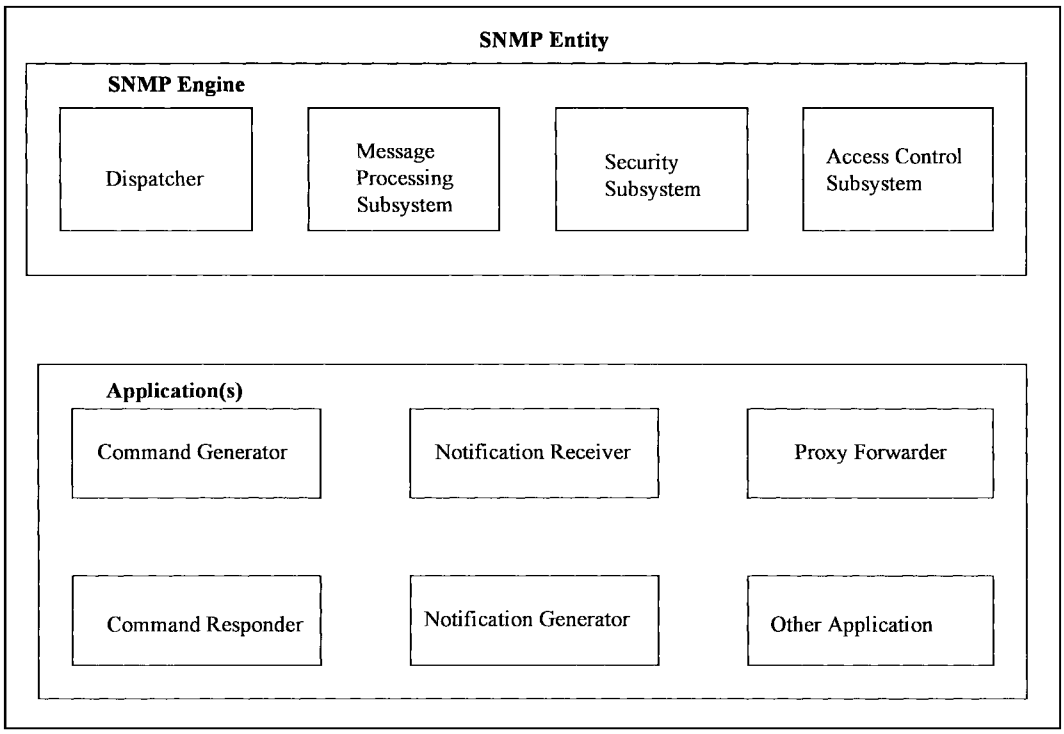


Figure 8-18
SNMPv3 architecture.

in Figure 8-18. The "Other Application" box in the figure is a placeholder for including other applications that may be developed in the future.

We now turn to some of the important concepts frequently used in SNMPv3.

8.9.1.1 contextEngineID. An *SNMP context* is a collection of objects accessible by an SNMP entity. As discussed, a collection of objects represents management information. A context can be a physical device, a logical device, multiple devices, a subset of a single device, or a subset of multiple devices. This definition of a context permits a certain degree of flexibility in grouping devices.

A contextName identifies a context; this contextName has to be unique within an SNMP entity. An SNMP entity can have one or more contexts. Again within an administrative domain, contextEngineID is used to represent an instance of a context. An administrative domain may have one or more contexts. Therefore, contextEngineID and contextName uniquely identify a context within an administrative domain.

The PDUs, along with contextEngineID and contextName, are encapsulated in a scoped PDU.

8.9.1.2 Principal. One of the important features of SNMPv3 security architecture is the concept of a *principal*. A principal uses the services provided by an SNMP engine. The SNMP engine does the processing for a principal. A principal can be one or more individuals, one or more applications, or a combination of individuals and applications. A principal is identified by a model-independent human readable string securityName that can be a global name. A model-dependent securityName is represented by a securityID.

8.9.1.3 SNMP Manager. The earlier concepts of an SNMP manager have been changed in SNMPv3. In the SNMPv3 architecture, an SNMP manager consists of one or more command generators, one notification receiver, and other components of an SNMP engine such as a dispatcher, message processing model, and transport mapper. Note that in SNMPv1 there is no concept of notification and a trap is used for sending unsolicited messages.

The interaction of a manager with the external components, along with the internal components of a manager, is shown in Figure 8-19. Though the RFC 2271 also shows a notification originator, we have excluded the notification originator, as a manager rarely issues a notification. However, it is possible that the notifications may be used in the future for some applications.

8.9.1.4 SNMP Agent. Just as the concept of a manager has been changed, the SNMPv3 architecture has also modified the way an SNMP agent is viewed. An SNMP agent contains one or more command responders and notification originators. The interaction of an agent in SNMPv3 with the components inside and the external managers is shown in Figure 8-20. Notice in the figure that command responders and notification originators interact with the MIB instrumentation.

8.9.2 SNMP Engine

An SNMP engine provides services for sending commands and responses in the form of messages, authenticating messages, encrypting messages, and providing access control to managed objects. The components of an SNMP engine are shown in Figure 8-21. Message processing subsystem and dispatcher are described in detail in RFC 2272 (Reference 8.13). RFC

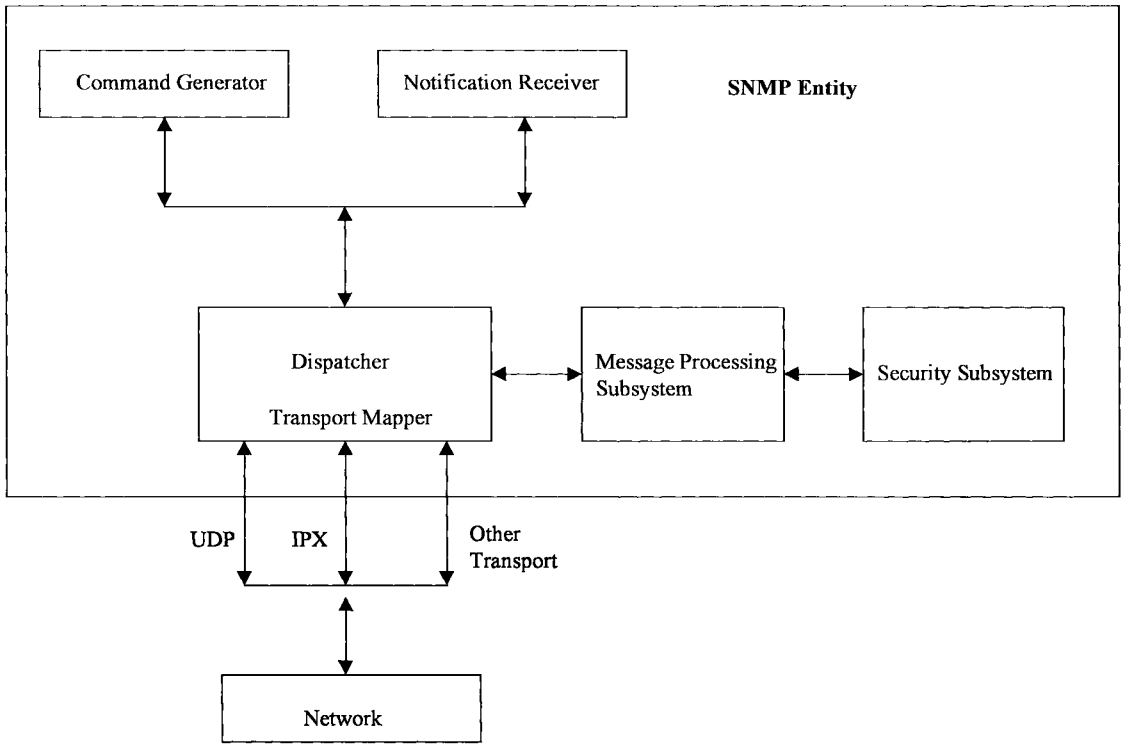


Figure 8-19
SNMPv3 manager.

2272 also defines the SNMPv3 message format and SNMPv3 message format furnished in Table 8-7.

We will briefly discuss each of the components of the SNMP engine.

8.9.2.1 Dispatcher. There is only one dispatcher for an SNMP entity. The dispatcher is similar in functions to the dispatchers used in operating systems and other areas. The dispatcher is primarily responsible for sending and receiving messages to and from networks that transport these messages. As SNMP has different versions, the dispatcher interprets different versions of SNMP and interacts with the appropriate message processing systems. It also provides interfaces to SNMP applications to send PDUs to an application or to a remote SNMP entity.

8.9.2.2 Message Processing Subsystem. The message processing subsystem is provided to account for different SNMP version messages. It aids

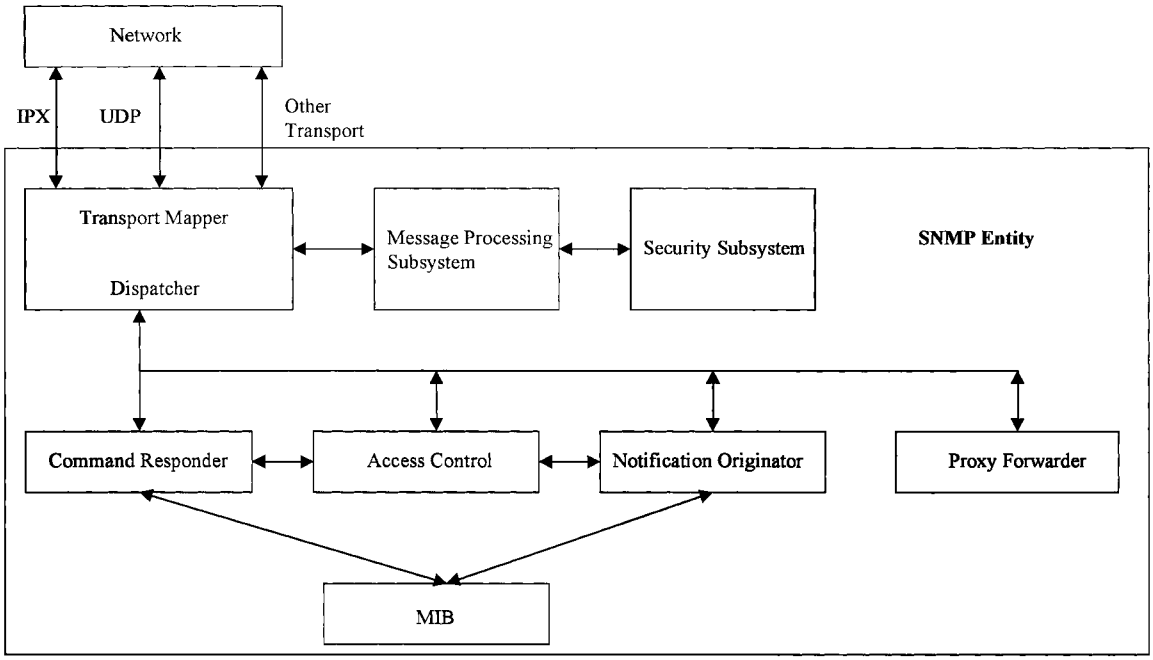


Figure 8-20
SNMPv3 agent.

in achieving interoperability with older SNMP versions and is responsible for preparing messages for delivery and interpretation and extraction of the data from received messages. These messages can be from or to SNMPv3, SNMPv1, SNMPv2c, or other message processing models.

8.9.2.3 Security Subsystem. The security subsystem includes one or more security models. It is responsible for providing security services such as authentication and privacy of messages, possibly by encryption. The definition of a security model is quite generic and is broad so as to include different security mechanisms to provide authentication and privacy. A security model includes threats against which security measures are provided, a security protocol, and the goals of the security model. A security protocol, in turn, includes mechanisms, procedures, and MIB data to provide authentication and privacy. A message processing model may control a security subsystem module.

8.9.2.4 Access Control Subsystem. The access control subsystem includes one or more access control models. Access control models include logic to determine whether the appropriate access rights are available.

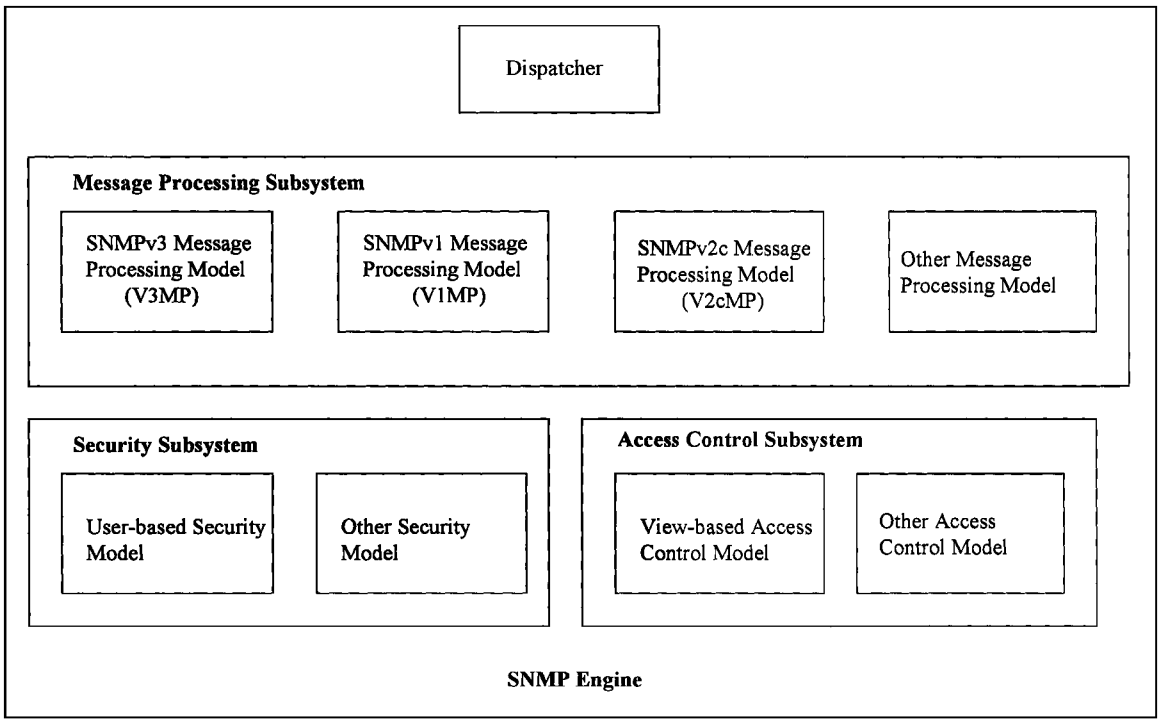


Figure 8-21
Components of an SNMP engine.

8.9.3 SNMPv3 Applications

RFC 2273 (Reference 8.14) explains SNMPv3 applications. It is essential to note that an SNMP engine does not have any restrictions on the type of applications that can be associated with it. As a result, an SNMP engine may be associated with a command generator, notification generator, or a proxy forwarder. We will now briefly look into each of the components of the applications.

8.9.3.1 Command Generator. A command generator performs the processing of requests and responses for the requests the command generator has initiated. The requests can be SNMP GetRequest, GetNextRequest, GetBulkRequest, or SetRequest.

8.9.3.2 Command Responder. A command responder receives requests such as SNMP GetRequest, GetNextRequest, GetBulkRequest, or SetRequest. It verifies whether the command is for the local system,

TABLE 8.7
SNMPv3 Message Format.

SNMPv3 Message Field	Details
msgVersion	INTEGER. For SNMPv3, value is 3.
msgGlobalData	msgID—INTEGER (0..2147483647); msgMaxSize—INTEGER (484..2147483647); msgFlags—OCTECT STRING (SIZE (1)); msgSecurityModel—INTEGER (0..2147483647). This field contains header and other administrative data.
msgSecurityParameters	OCTET STRING. This field is used by security model and the format of this field is defined by security model.
msgData	conextEngineID—OCTET STRING; contextName—OCTET STRING; data (RFC 1905 PDUs); encryptedPDU—OCTET STRING. This field can be unique context and plain text or encrypted data. If the data is encrypted, then it should be decrypted at the receiving end.

checks for access rights if mentioned in the request, performs the operation specified by the request, and generates an appropriate response to be sent to the initiator of the request.

8.9.3.3 Notification Receiver. A notification receiver listens to a received InformRequest PDU and generates responses to the InformRequest. As mentioned earlier, the InformRequest PDU is used for carrying a notification.

8.9.3.4 Notification Originator. A notification originator is responsible for generating a trap or InformRequest for a trap or notification, respectively, when certain events occur. The notification generator needs to know where to send the trap or notification, the SNMP version, and security parameters to use when generating a trap or InformRequest.

8.9.3.5 Proxy Forwarder. The definition of *proxy forwarder* is quite restrictive in SNMPv3. A proxy forwarder simply forwards SNMP requests, notifications, or responses without regard to their contents. Here no translations are involved in forwarding the messages. This explanation is slightly different from the explanation of the proxy furnished in Section 8.6. The implementation of a proxy forwarder is optional.

8.9.4 Abstract Service Interfaces and Primitives

Abstract service interfaces are abstract interfaces between subsystems in an SNMP entity. An abstract service interface includes a primitive and the

data elements to be passed when a service is invoked. Primitives define a set of services provided. The following primitives are defined for a dispatcher:

- *sendPDU:* An application uses this primitive to send an SNMP request or notification to a dispatcher.
- *processPDU:* A dispatcher sends this primitive to an application to process a PDU.
- *returnResponsePDU:* An application uses this primitive to send an SNMP response to a dispatcher.
- *processResponsePDU:* A dispatcher uses this primitive to send an incoming response PDU to an application.

An application requesting PDUs to be processed has to register with a dispatcher. For this registration, the primitive registerContentEngineID needs the parameters contextEngineID and pduType. pduType refers to one of the PDUs. When an application is ready to discontinue processing certain PDUs, it must deregister with the dispatcher. The primitive used for this is unregisterContextEngineID, and the parameters of unregisterContextEngineID are contextEngineID and pduType.

The following primitives have been defined for a message processing subsystem:

- *prepareOutgoingMessage:* Used for preparing an outgoing SNMP request or notification message.
- *prepareResponseMessage:* Used for preparing an outgoing SNMP response message.
- *prepareDataElements:* Useful for preparing abstract data elements from an incoming SNMP message.

The access control subsystem uses the isAccessAllowed primitive to check whether access is allowed.

Security subsystem uses the following primitives:

- *generateRequestMsg:* Used to generate a request or a notification message.
- *processIncomingMsg:* Used for processing an incoming message.
- *generateResponseMsg:* Used for generating a response message.

A common primitive, stateRelease, is used to release memory held. Figure 8-22 provides a diagram showing how a command generator or a notification originator can use the primitives to process a message to be sent out and the response to the message sent out. Similarly, the diagram in Fig-

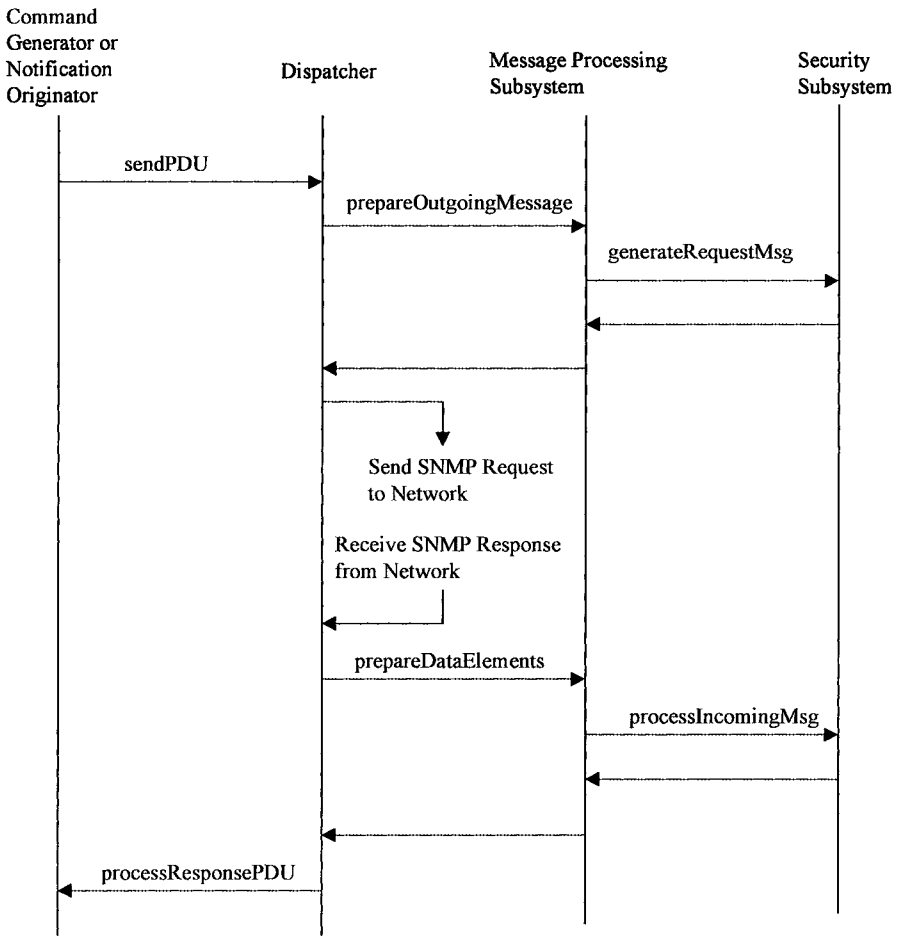


Figure 8-22 Command generator or notification originator scenarios.

ure 8-23 shows how a command responder or a notification receiver processes an incoming message and sends the response to the incoming message.

8.9.5 SNMPv3 Textual Conventions

SNMPv3 Architecture (Reference 8.12) defines the following SNMPv3 textual conventions:

- *SnmpEngineID:* Is an SNMP engine's unique identifier within an administrative domain. As an SNMP entity has only one SNMP engine, SnmpEngineID also uniquely identifies an SNMP entity. The syntax of SnmpEngineID is OCTET STRING (SIZE(1..32)).

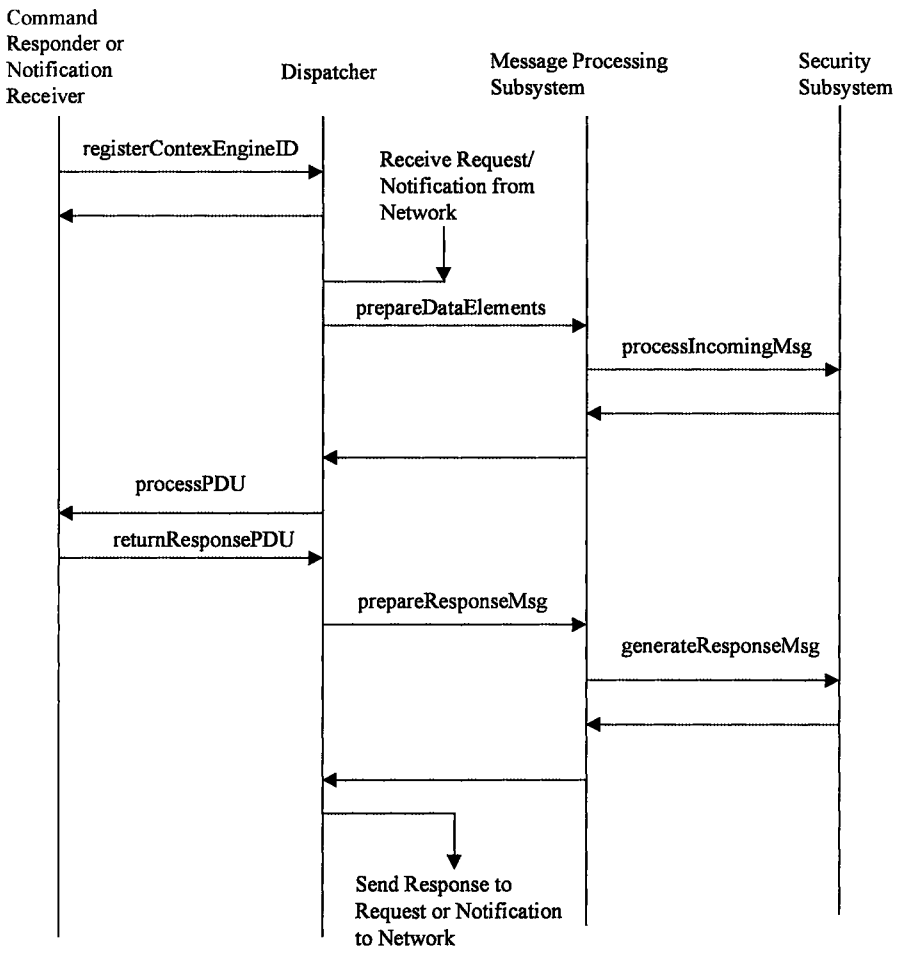


Figure 8-23 Command responder or notification receiver scenarios.

- *SnmpSecurityModel:* Is a unique identifier that identifies a security model within a security subsystem. The syntax is INTEGER (0..2147483647).
- *SnmpMessageProcessingModel:* Is an identifier to uniquely identify a message processing model within a message processing subsystem. Its syntax is INTEGER (0..2147483647).
- *SnmpSecurityLevel:* Indicates the security level for sending SNMP messages or processing messages. Three security levels have been defined: noAuthNoPriv (no authentication and no privacy), authNoPriv (authentication is used but there is no privacy), and authPriv (both authentication and privacy are used). The syntax of

SnmpSecurityLevel is INTEGER with values: noAuthNoPriv—1, authNoPriv—2, and authPriv—3.

- *SnmpAdminString:* Contains administrative information. It is better to have this octet string in human readable form. The syntax of SnmpAdminString is OCTET STRING (SIZE (0..255)).

8.9.6 User-Based Security Model

The user-based security model (USM) is discussed in RFC 2274 (Reference 8.15), which describes different authentication and privacy mechanisms that can be used. While the procedure for timeliness is fixed, there are provisions for using different algorithms for authentication and privacy. The user of a security model is the principal, discussed earlier.

The user-based security model uses the following three modules:

- *Authentication module:* Supports data integrity and data origin authentication.
- *Timeliness module:* Protects against message delay and message replay. A timeliness check is performed if there is an authentication check.
- *Privacy module:* Prevents unwarranted disclosure of a message. If privacy is selected, the architecture requires that authentication must also be performed.

We have already looked into some of the primitives provided by a security subsystem in Section 8.9.4. USM RFC introduces the following primitives:

- *authenticateOutGoingMsg:* Used to call the authentication module which implements a user's authentication protocol to authenticate a message before it is sent.
- *authenticateIncomingMsg:* Used to authenticate an incoming message.
- *encryptData:* Used to invoke privacy module to encrypt a serialized scopedPDU.
- *decryptData:* Used for unencrypting a received scopedPDU.

Each SNMP engine has the following three important objects:

- *snmpEngineID:* Is the unique identifier within an administrative domain of an SNMP engine.
- *snmpEngineBoots:* Refers to the number of times an SNMP engine has been rebooted or reinitialized since an snmpEngineID was last configured.

- *snmpEngineTime:* Refers to the time elapsed since snmpEngineBoots was reconfigured.

Let us look into how timeliness is checked. snmpEngineBoots and snmpEngineTime indicate the notion of time with respect to an SNMP engine. These are part of authenticated messages sent or received. These values are checked against a time window, which indicates the time within which a message has to be received. Thus a timeliness check ensures that message delay is within permissible limits.

The elements of the procedure used for preparing outgoing messages using authentication has been simplified and is shown in Figure 8-24. For more details on USM, refer to the RFC 2274 [8.15].

8.9.7 View-Based Access Control Model

RFC 2275 (Reference 8.16) describes the view-based access control model (VACM). The primary objective of the VACM is to check whether users (principals) have proper access rights to access one or more objects in a MIB and perform operations on these objects.

One of the key ideas in VACM is the notion of *MIB view.* A subset of a management information is known as MIB view. Access to a context occurs via a MIB view. For a context, there is normally one MIB view. Each MIB view has associated access rights of read-view, write-view, or notify-view.

The read-view access right permits reading values of object instances. Reading is done when retrievals such as GetRequest, GetNextRequest, or GetBulkRequest are done. Write-view allows the values of managed object instances to be changed by operations such as a SetRequest. Notify-view indicates the object instances that are permitted to send a trap or a notification.

Figure 8-25 shows how the primitive isAccessAllowed is used to check whether an application has the proper access rights to a set of object instances. For more details on the VACM, readers can consult Reference 8.16.

8.9.8 SNMPv3 MIB Modules

SNMPv3 requires additional objects besides those already defined in SNMPv1 and SNMPv2 because of the additional features such as security. There is also a need for defining more objects in the future, if additional

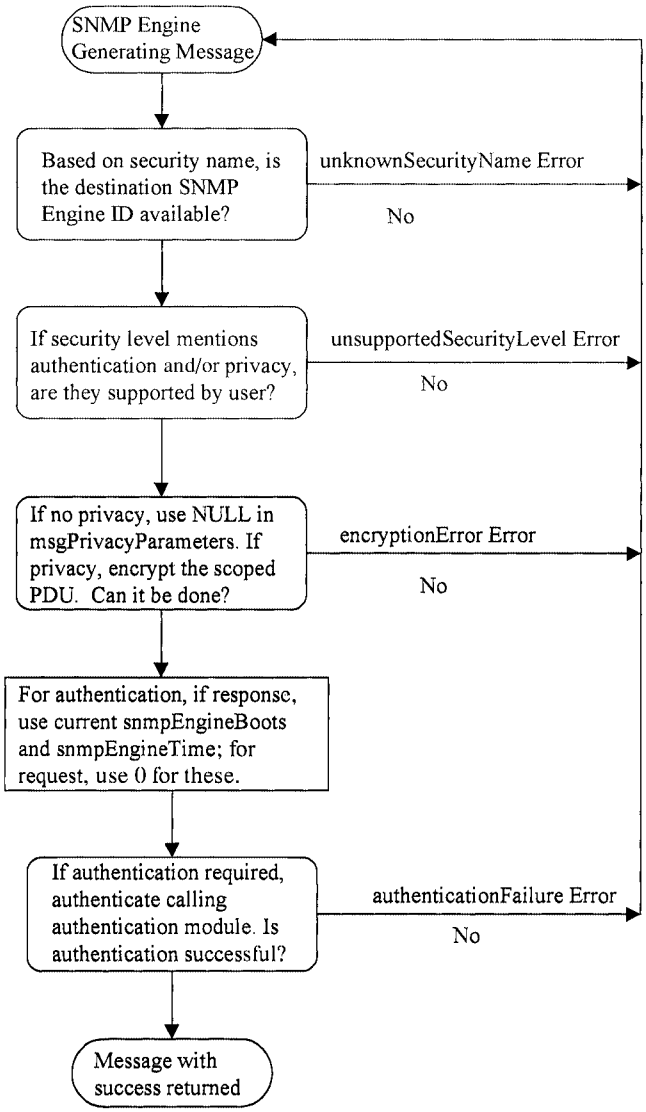


Figure 8-24 Generating an outgoing message.

functionalities for SNMPv3 are to be provided. For this reason, the administrative objects are left without objects below them. In the following discussion, we have not included MIB modules used for compliance and conformance of objects. The key object hierarchy of SNMPv3 is shown in Figure 8-26. The brief explanations of MIB modules are:

- *snmpFrameworkMIB:* Defines the MIB module for SNMPv3 architecture.

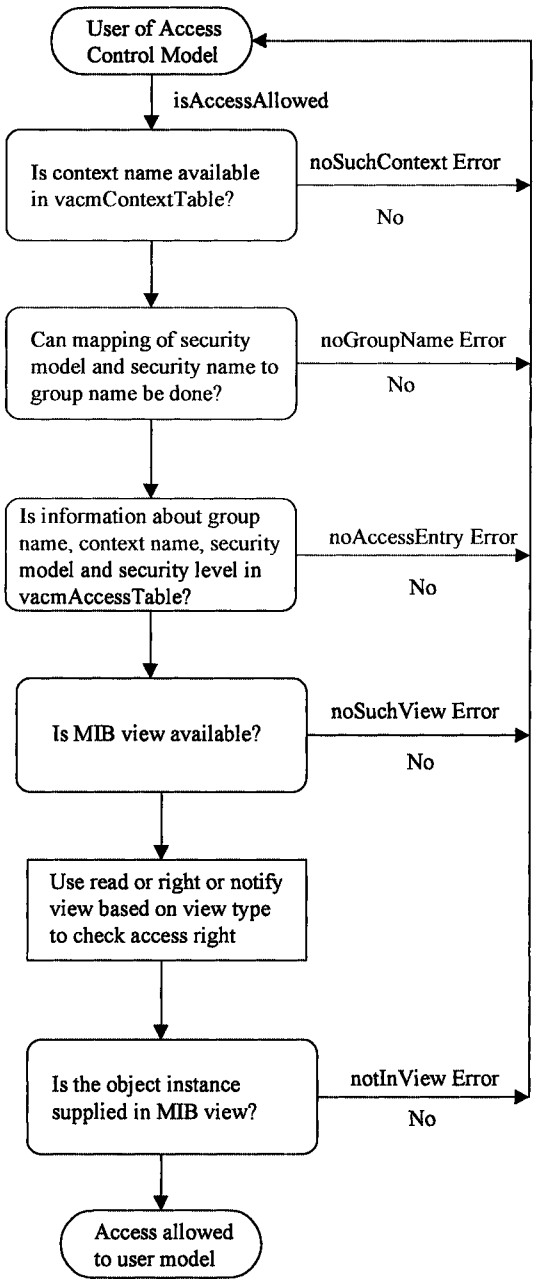


Figure 8-25
Processing of isAccess-Allowed service request.

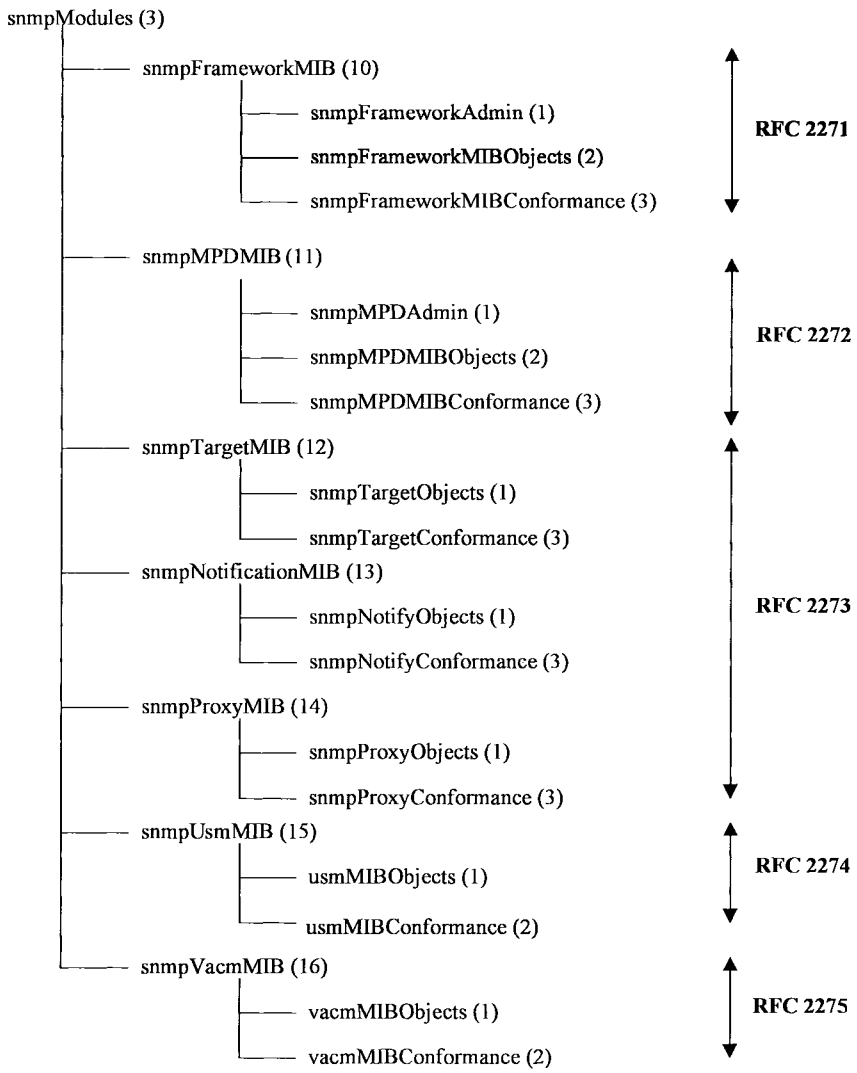


Figure 8-26 SNMPv3 MIB modules.

- *snmpFrameworkAdmin:* Used for the registration of SNMPv3 authentication and privacy protocol objects.
- *snmpFrameworkMIBObjects:* Used to derive SNMP engine-related objects such as snmpEngineID, snmpEngineBoots, snmpEngineTime, and snmpEngineMaxMessageSize.
- *snmpMPDMIB:* The MIB module for message processing and dispatching.
- *snmpMPDAdmin:* Used for the registration of message processing and dispatcher objects.

- *snmpMPDMIBObjects:* Provides a collection of objects for statistics on the number of packets dropped due to referencing the wrong security model, invalid or inconsistent components of the SNMP message, or no proper registration of SNMP application.
- *snmpTargetMIB:* Objects in this module are used for defining the destinations, transport domains, and addresses of the targets.
- *snmpTargetObjects:* Includes two tables, one that contains information about the transport domains and addresses and one that includes information about the SNMP version and security information for sending messages to specific transport domains.
- *snmpNotificationMIB:* Has objects for remotely controlling parameters of SNMP entities that generate notifications.
- *snmpNotifyObjects:* Determines the set of management targets that should receive notifications and the type notifications that should be sent. The objects in this group permit association of a set of filters with a target and restriction of the type of notifications for targets.
- *snmpProxyMIB:* Module objects are used to remotely configure the parameters in a proxy forwarding application.
- *snmpProxyObjects:* Provides objects for use by proxy forwarder applications.
- *snmpUSMMIB:* Includes objects used in the user-based security model.
- *usmMIBObjects:* Objects in this module are used to perform authentication and privacy on the management operations of users.
- *snmpVacmMIB:* Objects in this module are used for the view-based access control model.
- *vacmMIBObjects:* Contains objects that have details of the access rights available.

RFC 2273 defines three types of MIBs, namely SNMP-TARGET-MIB for management target, SNMP-NOTIFICATION-MIB for notifications, and SNMP-PROXY-MIB for forwarding proxy operations. Prior to RFC 2273, there were no specific MIBs to specify the destination or target MIBs. SNMP-TARGET-MIB and SNMP-NOTIFICATION-MIB permit the target, whether it is a trap or an inform, and the time and retry values for each target to be specified.

For event reporting, SNMPv3 protocol uses Trap and InformRequest. In SNMPv2, InformRequest was used for manager to manager communication in SNMPv2; its usage is slightly different in SNMPv3. Also, in SNMPv2 the use of Report PDUs was not clearly defined. Report PDUs in

SNMPv3 are used for communications between SNMP engines and are processed by message processing models.

8.10 Advantages of SNMP

As mentioned earlier, SNMP is a simple protocol, and it is easy to add this protocol to agents. Many applications using SNMP protocols have been implemented. It is easy to get and modify the SNMP code. Besides, the Internet is quite popular and has a large base. This in itself is a big advantage for SNMP.

Also, it is easy to add objects required for specific implementations. This can be done by adding the objects to the MIB. Continuous improvements and extensions are being added to the protocol with implementation experiences.

8.11 Notes on SNMP

SNMP centralizes network management activities in NMSs with minimum functionality in agents. This approach leads to scalability problems. The centralized model is suitable for small networks, but when large networks and heterogeneous networks are involved, there can be performance problems.

However, remote network monitoring (RMON) takes a decentralized approach to network management. RMON probes collect data from the individual Ethernet segments or token ring networks, and perform processing and sorting locally and forward the data to the NMS using SNMP. Some of these issues, such as centralized management and decentralized management, are also treated well in Reference 8.19.

As we move into a client-server environment with a large number of networks, management domains and communications between them will assume a significant role. Management domains are also important for distributed network management. Incidentally, there is much work to be done in the Internet network management arena. Security is very critical in distributed environments, but currently this issue is not being addressed. This hole in SNMP security has to be addressed.

Common object request broker architecture (CORBA) is also becoming very popular in distributed network management. Therefore, the interoperability between CORBA and SNMP has to be worked out to utilize the

power of CORBA. In the Internet, the distributed management (DISMAN) group is working on the distributed aspects of network management. This group should address some of the issues just mentioned.

When SNMP was designed, the idea was to keep the management protocol simple. This has its own drawbacks, however. The network management has a whole range of functions in addition to the SNMP protocols defined. An Internet network management framework for network management applications similar to systems management functional areas of the OSI needs to be delineated to standardize this area also.

It is also necessary to define objects for the different network management functions such as configuration management, fault management, security management, performance management, and accounting management. These may be borrowed from OSI. This is required as the convergence of network management of ITU-T and the Internet may not happen at all.

In configuration and fault management, relationships between different monitored objects are very important in tracing the origins of faults. Thus, relationships must be defined and standardized.

In configuration management, aggregation objects need to be defined. In some cases, the status of an object may affect the aggregation of objects below a subtree. In such cases, the status will have to be combined by either simple or weighted aggregation. Also, the configuration may be a combination of objects; for example, a network may be a combination of an IP network, telecommunications network, and a network using a proprietary protocol. To show the whole network, it is necessary to define aggregate objects.

The Internet group is focused on protocols used by the Internet community, and it expects a certain layer stack. It also assumes certain physical layer interfaces. As an example, SNMP protocols place emphasis on Ethernet LANs and the TCP/IP protocol suite. But there are protocols, such as token ring, which are also popular. The focus on Internet-related protocols may be regarded as a strength as well as a weakness. It is a strength in that problems facing the Internet community get immediate attention, but a weakness in that others may have to wait longer for standardization.

Therefore objects may have to be specifically defined and extensions added to the MIB for many resources. This may result in defining the same resources in different manners, which can create problems in the interoperability of network management functions.

The definition of managed object classes in ITU-T allows the use of inheritance and polymorphism. This makes possible the reuse of attributes; hence the implementations become simpler with this approach. We have already explained polymorphism in Chapter 4, Section 4.14. Poly-

morphism helps in the migration and coexistence of different versions of the same applications.

We have seen in the ITU-T that managed objects have attributes, operations, notifications, and behaviors. In SNMP, object definitions are less powerful and carry less information; this becomes a problem in some cases where performance is an important criterion.

In SNMP protocols, there are no clear-cut rules regarding how to partition MIB-II objects into different systems management functional areas. Unless there is standardization in this direction, the partitioning of MIB-II objects will not be uniform in implementations from different vendors. In addition to the MIB-II objects, there should be well-laid-out rules for partitioning the device- and protocol-specific objects into different systems management functional areas.

In TMN, ITU-T standards are widely used. In the computer industry, the standards are not so rigid, with leading computer vendors setting the trend on the "standards." Because of the simplicity of SNMP, SNMPv1 is popular and is attractive to device vendors doing instrumentation. To support or extend new device features, MIBs have to be extended. This can be done easily. As a result of these strong points of SNMP, there are also implementations based on SNMP in the TMN arena.

However, one of the major problems with SNMP and the Internet RFCs is that they are frequently changed or updated. Sometimes this can create problems for implementers. SNMPv2 has not made much headway in the industry because of the complexity and controversies. We have alluded earlier to the fact that SNMPv3 RFCs have many outstanding issues to resolve. To address these issues, it is expected that updated RFCs will replace the existing SNMPv3 RFCs.

Besides, SNMP is just a management protocol. Network management involves more than the management protocols. The success of network management will depend primarily upon the ease of use and the variety and usefulness of management applications. As far as SNMP is concerned, a lot of work needs to be done to standardize management applications as in OSI systems management, TMN, distributed network management, interoperability with CORBA, and so on.

Without much standardization in these areas, it is difficult for the vendors to develop management applications. Development of management applications is very important if SNMP is to get much of a foothold in TMN. Besides, some management applications have a long life, and few people are willing to replace or change these management applications. So, caution is advised before adopting SNMP in TMN.

8.12 Coexistence Between SNMP and CMIP

Initially, while Internet network management protocols were being developed, the intention of SNMP was to merge eventually with ITU-T and OSI management protocols. SNMPv1 got a boost and some following in the industry because of the simplicity of implementation and its value in small applications. However, due to the existence of a large base of SNMP managed devices, there is no effort to merge SNMP and CMIP.

The telecommunications management models are more appropriate for transmission systems and cross-connections than for packet networks such as those used in the Internet. In the Internet, packets may be routed differently at different times.

CMIP is generally regarded as complex and is good for large systems. SNMP is simple and good for small systems. In CMIP, managed objects can contain other managed objects or point to other managed objects using attributes. In SNMP, the objects can point to other objects and collection of data is limited to data on attributes and behavior descriptions. As a result, in reality, there is a need for coexistence of the CMIP and SNMP management protocols. TeleManagement Forum has addressed the issue of interoperability between CMIP and SNMP.

8.13 Interoperability with ITU-T/OSI

For interoperability between ITU-T/OSI and TCP/IP protocols, RFC 1006 is recommended. RFC 1006 mimics the transport layer on top of TCP as shown in Figure 8-27. In this way, layers such as the session, presentation, and application layers can run on top of TCP/IP protocols. This enables applications such as TMN and mail services to operate on TCP/IP. Some of the salient features of RFC 1006 are:

- *ISO transport Class 0 is supported.* Class 0 provides the most simple basic connection-oriented support during connection establishment and connection release phases. It does not provide a mechanism to detect protocol errors. Quality of service, which classifies transport services in terms of Class 0, Class 1, Class 2, Class 3, and Class 4, is not supported.

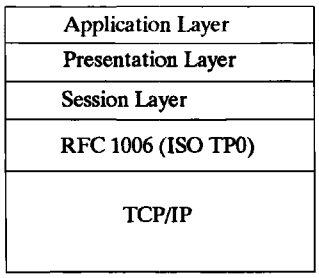


Figure 8-27
Interoperability with OSI.

- *Some departures from Class 0 are made.* Initial data may be exchanged during the connection establishment phase; expedited data service is supported; and much larger transport protocol data unit (TPDU) size is supported. For performance reasons, a default TPDU size of 65531 is taken. However, smaller TPDU sizes are allowed, but they must be negotiated.
- *Network service is provided by the TCP.*
- *TPDU is encapsulated within each packet designed specifically for this RFC.* This packet consists of a packet header and the TPDU. The packet header has 8 bits for version, 8 bits of reserved field, and 16 bits of length field.

8.14 Implementation Notes

TCP/IP protocol suite RFCs, including those of network management, change quite frequently to meet new requirements. Before any RFC is implemented, it must be ascertained that it is the latest and is not superseded by a more recent RFC. To do so, the first document to consult is the *RFCINDEX*, which is an up-to-date index of the RFCs that have been released. The next document to consult during design and implementation is the RFC on *IAB Official Protocol Standards*, which furnishes the maturity levels of RFCs. These documents are regularly updated.

Also, for implementation purposes, informational, experimental, and historical states are not of much use. By and large, it is better to go for full-scale implementation after an RFC has reached at least the draft stage.

8.15 Internet Standardization Process

In the Internet community, topics of interest are written as requests for comments (RFCs). There are standard formats for writing RFCs. There are two separate tracks for RFCs: standard and nonstandard. Nonstandard RFCs include experimental, informational, and historic RFCs. A document that is not ready for standard track is delegated to the experimental stage. An RFC that is replaced by some other document or whose contents are obsolete enters the historical state.

The interests, activities, and development of standards for the Internet are controlled by the Internet Activities Board (IAB). The IAB is subdivided into different functional subgroups. One of the subgroups, the Internet Engineering Task Force (IETF), is involved in protocol development and standardization activities. The main objective of the IETF is to resolve any issues that may crop up and come up with solutions, standards, and architectures for short- and mid-term protocols. Another subgroup, the Internet Engineering Steering Group (IESG), studies RFCs and recommends movements of standards to different maturity levels.

The RFCs in the standard track enter different maturity levels in stages and time frames. When a specification is developed with the intent of moving it in the standard track, it can be termed a *prototype*. An RFC starts as a prototype and moves to the proposed state, advances to a draft standard state, and then becomes a full standard. When an RFC advances to the next maturity levels, rigorous review is done. Also, implementation and interoperability are required for moving from the proposed to the draft standard stage. At any stage, an RFC can be moved to different states in the nonstandard track. The Internet standard process is explained in RFC 2026 (Reference 8.17).

8.16 Summary

In this chapter, we have examined the basic philosophy of Internet network management and introduced the SNMP management protocols framework. We started the chapter with a detailed discussion of SNMPv1. As SNMPv1 is very popular in the computer industry, and, to a lesser degree, in the telecommunications industry, it is essential to look into the basic concepts of SNMPv1 protocols. We next discussed the SNMPv2 and the RFCs associated with SNMPv2, which are draft standards. SNMPv3, which is an extension SNMPv2, is discussed briefly. We have also looked

into the topics of interoperability with OSI/ITU-T, as well as the advantages and limitations of SNMP protocols. We end this chapter with an overview of the Internet standardization process for a better understanding of the different versions of SNMP protocols and their standing with respect to the Internet standard track.

8.17 References

8.1. McCloghrie, K. and M. T. Rose, Structure and Identification of Management Information for TCP/IP-based Internets, RFC 1155, 1990.

8.2. Case, J. D., M. Fedor, M. L. Schoffstall, and C. Davin, Simple Network Management Protocol, RFC 1157, 1990.

8.3. Rose, M. T. and K. McCloghrie, Concise MIB Definitions, RFC 1212, 1991.

8.4. McCloghrie, K. and M. T. Rose, Management Information Base for Network Management of TCP/IP-based Internets: MIB-II. RFC 1213, 1991.

8.5. Case, J., K. McCloghrie, M. Rose, and S. Waldbusser, Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1902, 1996.

8.6. Case, J., K. McCloghrie, M. Rose, and S. Waldbusser, Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1903, 1996.

8.7. Case, J., K. McCloghrie, M. Rose, and S. Waldbusser, Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1904, 1996.

8.8. Case, J., K. McCloghrie, M. Rose, and S. Waldbusser, Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1905, 1996.

8.9. Case, J., K. McCloghrie, M. Rose, and S. Waldbusser, Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1906, 1996.

8.10. Case, J., K. McCloghrie, M. Rose, and S. Waldbusser, Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1907, 1996.

8.11. Case, J., K. McCloghrie, M. Rose, and S. Waldbusser, Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework, RFC 1908, 1996.

8.12. Harrington, D., R. Presuhn, and B. Wijnen, An Architecture for Describing SNMP Management Frameworks, RFC 2271, 1998.

8.13. Case, J., D. Harrington, R. Presuhn, and B. Wijnen, Message Processing and Dispatching for the Simple Network Management Protocol (SNMP), RFC 2272, 1998.

8.14. Levi, D., P. Meyer, and B. Stewart, SNMPv3 Applications, RFC 2273, 1998.

8.15. Blumenthal, U. and B. Wijnen, User-based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3), RFC 2274, 1998.

8.16. Wijnen, B., R. Presuhn, and K. McCloghrie, View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), RFC 2275, 1998.

8.17. Bradner, S., The Internet Standards Process—Revision 3, RFC 2026, 1998.

8.18. Perkins, D. T., "Questions Answered," *The Simple Times*, vol. 6, no. 1, pp. 17—21, 1998.

8.19. Meyer, K., M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt, and Y. Yemini, *Decentralizing Control and Intelligence in Network Management.* Integrated Network Management IV, Proceedings of the Fourth International Symposium on Integrated Network Management, 1995, Sethi, A. S., Raynaud, Y., and Faure-Vincent, F. (eds.), London: Chapman & Hall, pp. 4—15, 1995.

8.20. McGloghrie, K., "SNMP Framework," *The Simple Times*, vol. 4, no. 2, pp. 23—24, 1996.

8.21. SNMP Research, Inc., Is SNMPv2 Really Dead? 1995. Also visit the SNMP Research SNMPv2* Web Site.

8.22. McGloghrie, K., "SNMP Framework," *The Simple Times*, vol. 4, no. 3, pp. 21—22, 1996.

8.18 Further Reading

8.1. Rose, M. T., *The Simple Book: An Introduction to Internet Management* (2d ed.), Englewood Cliffs, NJ: Prentice Hall, 1996 (1st ed. published 1991).

8.2. Stallings, W., *SNMP, SNMPv2, and RMON: The Practical Network Management Standards*, (2d ed.), Reading, MA: Addison-Wesley, 1996. At press time, this book was being revised to include SNMPv3 and RMON2.

8.3. Postel, J., Transmission Control Protocol, RFC 793, 1981.

8.4. Leinwald, A., and K. Fang, *Network Management: A Practical Perspective*, (2d Edition) Reading, MA: Addison-Wesley, 1995.

This page intentionally left blank.

PART 3

TMN Applications

This page intentionally left blank.

CHAPTER 9

Network Management for Mobile Communications

9.1 Introduction

Because the subject of this book is TMN, we will only discuss abstractly the different forms of communications technologies. Those who are interested in the specifics of these technologies should refer to one of the many good books available on the subject.

Communication between any devices can be classified as wired or wireless (Figure 9-1). In wired communication, conductors such as copper twisted pair, fiber optics, or coaxial cable carry information between two or more devices, whereas in wireless communication, space is the communication media.

Another classification of communication is based on the type of connection between an end user and network services. In fixed communication, there is a permanent association between an end user device and a single network access point. In mobile communication, however, an end user device can connect to multiple access points and use network services for communication between end-user devices. Mobile communication consists of portable and frequently moving devices. A mobile communication network includes personal communications services (PCS) networks, wireless LAN networks, and satellite networks.

PCS is a wireless technology based on digital communications. Cordless phones and cellular phones are early versions of the PCS. For PCS, two digital technologies—time division multiple access (TDMA) and code division multiple access (CDMA) are popular protocols. For TDMA, there are again two standards: North American standards based on TIA/EIA/IS-136, and Digital Cellular System 1800 (DCS 1800) standards developed by the European Telecommunications Standards Institute (ETSI).

Wireless communication is increasing in market penetration. For TMN of PCS, ANSI standards are available. In European nations, ETSI global system for mobile communication (GSM) standards are quite popular. Bellcore has also developed its own TMN documents for wireless communications.

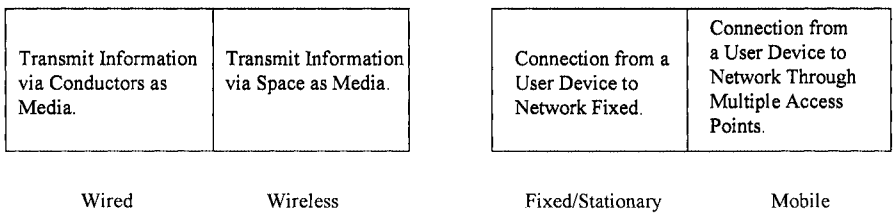


Figure 9-1
Differences between communication terminologies.

9.2 Overview of Network Management for Mobile Communications

Network management for mobile networks is more complex than network management for fixed or stationary networks. Some of the important issues involved in the network management of mobile networks are:

- The design of MIBs has to satisfy the requirements of dynamic topology changes; operations under stress; and interactions with different physical environments including rugged physical factors such as varying weather conditions, electromagnetic interference, and multipath effects.
- Mobile networks operate under operational stress due to the limited bandwidth available. Mobile networks are operated near capacity limits, unlike stationary networks, which are normally operated below capacity limits. This causes problems in distinguishing stress conditions from failure conditions. Operational stress also creates problems when alarm correlation is used to capture the root cause of a problem. One solution is to partition a network into smaller domains. These smaller domains constrain alarm correlation and problem diagnosis to a manageable level. Reference 9.15 treats this issue thoroughly.
- Network topology in mobile networks can frequently change because of roaming and handoff, so element managers must be able to handle dynamic topology changes. In fixed networks, topology changes are normally minimal.
- Mobility effects billing and routing. As an example, there must be coordination between the visiting domain and the home domain during roaming. To support roaming, routing functions must track the locations of mobile stations (MSs). A cellular phone is an example of an MS. Similarly, the billing function must be able to correctly collect and share the billing information among the visiting and home domains.
- At the service level, managing QOS parameters is more complicated. In mobile networks, there are QOS parameters such as connection setup time and call losses.
- Provisioning is more complex in mobile networks than in stationary networks. End-to-end communication involves mapping of names to addresses and addresses to routes. These are maintained in different

configuration databases. Access authorization, accounting, and service parameters depend upon the data maintained in the configuration databases. In the case of mobile networks, as the changes are dynamic, the data in these configuration databases dynamically change. This entails keeping track of rapid changes in databases as well as maintaining consistency.

CMIP and SNMP are popular management protocols for mobile communications. We have already covered these management protocols in Chapters 7 and 8.

One of the challenges faced in network management of mobile communications is the variety of vendor services and ownerships available. As an example, in a PCS network, one vendor may own both the radio and switching resources. In another model, one vendor may own radio services and another may own the switching services. This leads to the challenge of managing hybrid modes of operation. Even though there are multiple service providers, these service providers must be able to share data related to billing, security, and subscriber profiles.

9.3 ANSI Network Management for PCS

A brief introduction to management mobility application protocol (MMAP) is furnished to make readers aware of the management capabilities available for some components of PCS. ANSI T1.651-1996 (Reference 9.13) defines the MMAP for exchange of information and invocation of operations to support terminal mobility in a wireless environment. MMAP is useful for providing functions such as call waiting, call control, call setup, call manipulation, call clearing, registration, location updating, authentication, roaming, and handover. Extensions and revisions to ANSI T1.651-1996 are furnished in ANSI T1.651a-1996 (Reference 9.14). MMAP protocol is an application layer protocol to be used between radio systems and network elements such as mobility management platforms, switching systems, and other radio systems. For more details on MMAP, consult References 9.13 and 9.14.

ANSI T1.244-1995 (Reference 9.11) contains details of information model and OAM&P requirements. This standard is an extension to ANSI T1.210 and ITU-T Recommendation M.3010, Principles for a Telecommunication Management Network.

A Q3 interface is used between an OS and network elements representing different functional elements used in PCS. Functional elements of a PCS network are:

- *Radio Access System Controller (RASC):* Is used to manage terminal mobility.
- *Radio Port Controller (RPC):* Provides access to wireless components.
- *Terminal Mobility Controller (TMC):* Provides terminal mobility within a PCS network.
- *Terminal Mobility Datastore (TMD):* Provides storage for the data needed for terminal mobility.
- *Personal Mobility Controller (PMC):* Is used for controlling personal mobility.
- *Personal Mobility Datastore (PMD):* Contains data on personal profiles.
- *PCS Switching Center (PSC):* Connects and routes user traffic from a wireless network to a wireline or other wireless networks.

These functional elements are described in Reference 9.12. There are no well-defined rules for mapping these functional units to network elements. A network element may represent one or more functional units. ANSI T1.244 defines the information model for RASC and RPC only. Many object classes are drawn and derived from ITU-T Recommendation M.3100, Generic Network Information Model, and ITU-T Recommendation X.721, Structure of Management Information: Definition of Management Information.

Network management for PCS uses the following principles laid out in TMN architecture:

- A logical layered architecture is used that contains five layers, namely business management, service management, network management, element management, and network elements.
- The physical architecture defines operations systems, communication, networks, and network elements.
- The functional architecture is similar to that defined in TMN M3010. Refer to Chapter 2, Section 2.3.1 of this book for details.
- SMFAs such as configuration management, fault management, performance management, accounting management, and security management are used.

ANSI T1.244 covers fault management and configuration management function areas in detail, as well as requirements for PCS management

function areas such as configuration management, fault management, performance management, accounting management, and security management. Some of the ITU-T recommendations that are useful for network management of PCS are as follows:

- *Configuration Management:* Uses the services and functions defined in X.730, Systems Management: Object Management Function.
- *Alarm Surveillance:* Uses the services and functions defined in Q.821, Stage 2, and Stage 3 Description for the Q3 Interface, Alarm Surveillance.
- *Failure Localization:* Needs the services and functions defined in X.745, Systems Management: Test Management Function, and X.737, Systems Management: Confidence and Diagnostic Test Categories.
- *Testing:* Employs the services and functions defined in X.745, Systems Management: Test Management Function, and X.737, Systems Management: Confidence and Diagnostic Test Categories.
- *Performance Monitoring:* Uses services and functions defined in Q.822, Stage 1, Stage 2, and Stage 3 Description for the Q3 Interface, Performance Management.

9.4 ETSI Network Management for DCS

GSM standards developed by ETSI are standards accepted in many countries for digital cellular systems. GSM standards have been developed in three phases. Phase 1 standards are devoted to the introduction of GSM services. Phase 2 covers enhancements for Phase 1. The next phase, known as Phase 2+, includes enhancements such as voice coding and improved data transmission services.

9.4.1 GSM Architecture

The GSM system architecture is shown in Figure 9-2. We will discuss the GSM architecture (Reference 9.10) as it is relevant to network management.

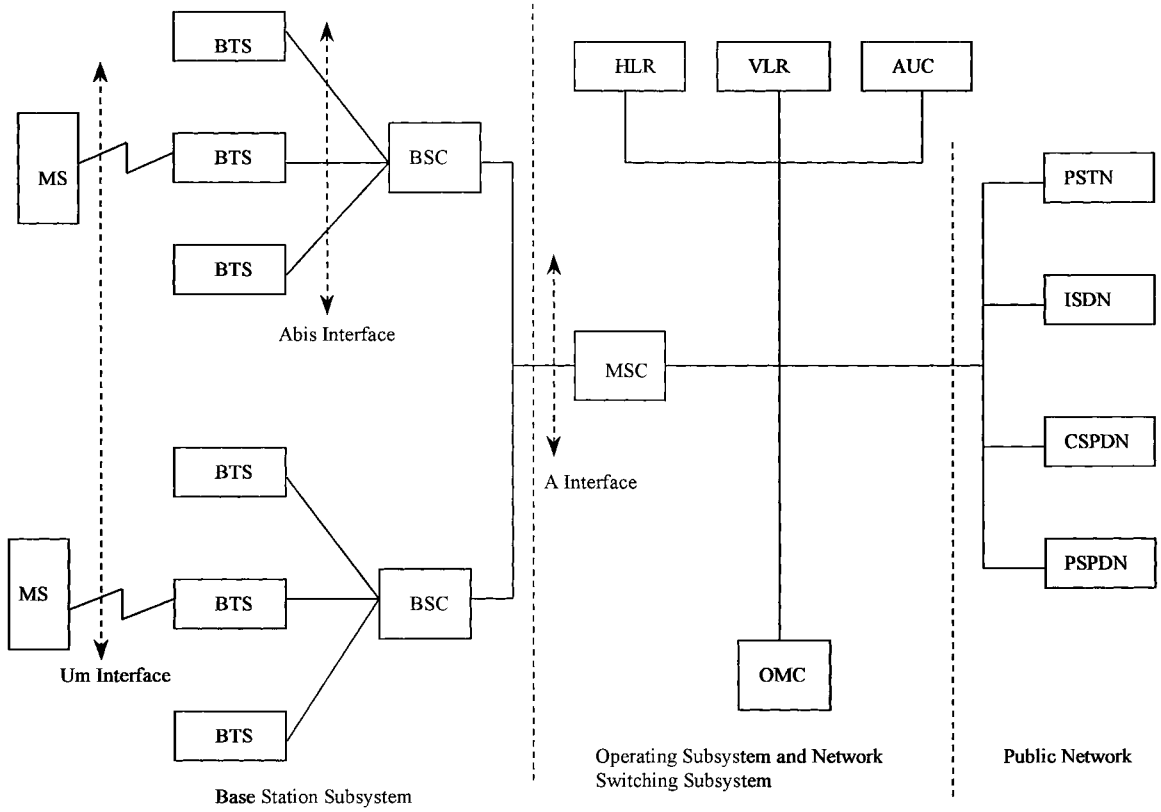


Figure 9-2
GSM architecture (© 1997 IEEE).

There are basically three interfaces: the air interface (Um), Abis, and A, as shown in Figure 9-2. A base station subsystem (BSS) includes a base transceiver station (BTS) and a base station controller (BSC). The BTS handles radio interfaces with mobile stations (MSs), and the BSC manages the handovers and radio resources. As can be seen from the figure, a BSC can manage one or more BTSs.

A mobile switching center (MSC) is responsible for communication with other networks such as public switched telephone networks (PSTNs), ISDNs, circuit-switched public data networks (CSPDNs), and packet-switched public data networks. There are three databases, namely home location register (HLR), visitor location register (VLR), and authentication center (AUC).

9.4.2 Overview of Network Management of PLMN

ETSI has developed GSM 01 thru 11 series standards for implementing a public land mobile network (PLMN). Note that European Telecommunications Standard (ETS) and GSM standards are identical. As an example, GSM 12.00 (Reference 9.1) is also named ETS 300 612-1.

ETSI GSM 12 series standards cover management of PLMN components; interconnectivity of management components; and operations, administration, and maintenance (OAM). Basically, GSM 12 series standards extend the ITU-T TMN standards and ISO standards to the PLMN network management standards. We have included some ISO standards in the discussion, as some of the ISO standards do not have equivalent ITU-T standards.

As stated in Reference 9.1, "a PLMN is a telecommunications system consisting of several functional units necessary to perform mobile telecommunications services." A PLMN consists of different mobile equipment such as location registers, mobile services switching centers (MSCs), base station systems (BSSs), mobile stations (MSs), authentication center (AUCs), transcoders, transmission equipment, echo suppression equipment, and so on. As the focus of the chapter is on network management for mobile communications, we will not look into each component of PLMN. For more information on these, refer to 0 through 11 series GSM standards. The management of a PLMN includes the existing terrestrial telecommunications and the extensions to cover mobile telephony.

The scope of network management of PLMN covers integrated operation of PLMN components, inter-PLMN operation of the mobile systems, and delivery of the promised QOS. The management PLMN should also support service and business areas. Service and business areas are not included in the 12 series standards. However, they are important as they are related to management of PLMN. Services and business areas perform the following functions:

- *Administration of Subscribers:* Is concerned with connecting and disconnecting services of a subscriber, testing the services offered, and upgrading the services available to include providing additional services and new services when they become available.
- *Collection of Charges from Subscribers:* Involves efficient and accurate recording of billing amounts from subscribers and resolution of billing discrepancies.

- *Collection of Revenue from Other Operators:* Deals with issues involved in intercompany billing. It is necessary to have a precise division of revenue between companies to avoid conflicts.
- *Maximization of Revenue from Network Resources:* Has to do with optimum utilization of resources, for example, maximizing operational revenues.
- *Customer Services:* Includes services such as repair, directory assistance, billing details, assistance in using service, and others. These services can generate additional revenues and are important to keep subscribers satisfied.

9.4.3 Operation, Administration, and Maintenance (OAM)

Operation, administration, and maintenance (OAM) is different from OSI SMFAs, which we have considered in Chapter 1, Section 1.6. OAM is also not as exhaustive and complete as the TMN management services defined in M.3200. OAM components are:

- *Administration and Commercial:* Includes functions for collecting data that can be used for billing, such as management of subscribers and collection of subscriber data, mobile equipment data, and call data. These functions are initiated by an administration center (ADC). ADC control is within the TMN. These functions are further explained in GSM 12.02 (Reference 9.4) and GSM 12.05 (Reference 9.7).
- *Security:* Is based upon the responsibilities to be enforced. Subscriber access and usage of services must be permitted only to authorized persons. Similarly, security of TMN data, especially billing-related data, must be maintained. There is also a provision for security audit trails to enable any security violations to be studied in detail. The security function area is explained in GSM 12.03 (Reference 9.5).
- *Operations and Performance:* Objectives are to enable an operator to monitor performance and, if required, to permit the operator to alter the configuration to improve performance and QOS provided to subscribers. To do this, it is necessary to collect the right data at the right time for analysis and display of performance-related data. For this it may be necessary to execute traffic management commands and functions. Operations and performance-related functions are treated in GSM 12.04 (Reference 9.6) and 12.06 (Reference 9.8).

- *Change:* Related to functions that minimize the impact of changes on the integrity and stability of PLMNs. Changes include logging of data and changes in data. This function is described in GSM 12.06 (Reference 9.8).
- *Maintenance:* An important activity in delivery of the agreed-upon QOS. There are proactive and reactive maintenance activities: Proactive maintenance includes routine maintenance to detect the problems ahead; reactive maintenance is done after the fact or after the occurrence of a problem. A problem is noticed when a failure message in the form of an alarm, customer complaint, or an alert caused by thresholds is received. Maintenance is covered in GSM 12.11 (Reference 9.9).

Some of the maintenance functions are:

- *Monitor:* Includes monitoring the network for alarms or changes in operational trends. Some of the operational trends can be detected by alerts generated by crossing of thresholds.
- *Detect:* The detection of a problem includes an effective reporting mechanism. Sometimes detection of a problem can be linked to trouble ticket administration.
- *Localize:* Once a problem is reported, the cause of the failure is isolated to enable corrective action.
- *Rectify:* Deals with solving a previously reported problem.
- *Restore:* Involves making the service or services a customer was using available again after a problem is corrected.
- *Record:* Aids in recording a failure, its causes, and how the failure was rectified. This record keeping helps in solving similar problems and in analyzing problems at a later date.

9.4.4 PLMN Information Model

Modeling is an important activity in the formation of information models. Modeling provides abstract representation of the resources and the management actions that can be performed on these resources. In GSM, as in TMN, object-oriented techniques are widely used for the definition of managed object classes. GDMO and ASN.1 definitions use object-oriented principles. The inheritance hierarchy of managed object classes defined for GSM is shown in Figure 9-3. The containment hierarchy of the managed object classes defined in GSM series documents is shown in

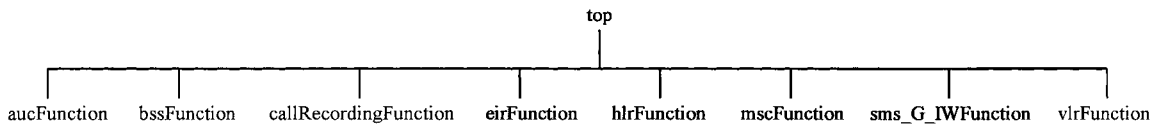


Figure 9-3
Inheritance hierarchy of GSM object classes.

Figure 9-4. In this figure, a network is imported from M.3100 and represents the identifier of the telecommunications network. plmnNetwork represents the characteristics of the PLMN. managedElement represents an NE and is the starting point for modeling GSM-specific managed object classes. In addition to the MOCs defined in the GSM series, some of the MOCs defined in M.3100 are reused during formation of the information model for GSM.

GSM 12 series standards have defined PLMN-specific managed object classes. Their short descriptions are as given below:

- *aucFunction:* Includes all functions required to implement AUC.
- *bssFunction:* Includes all functions required to implement network element BSS.
- *callRecordingFunction:* Is used to model the mechanism and controls required to generate and collect data for a network usage.
- *eirFunction:* Is used to model the equipment identity register (EIR).
- *hlrFunction:* Represents the resource HLR.
- *mscFunction:* Includes all functions required to implement MSC.

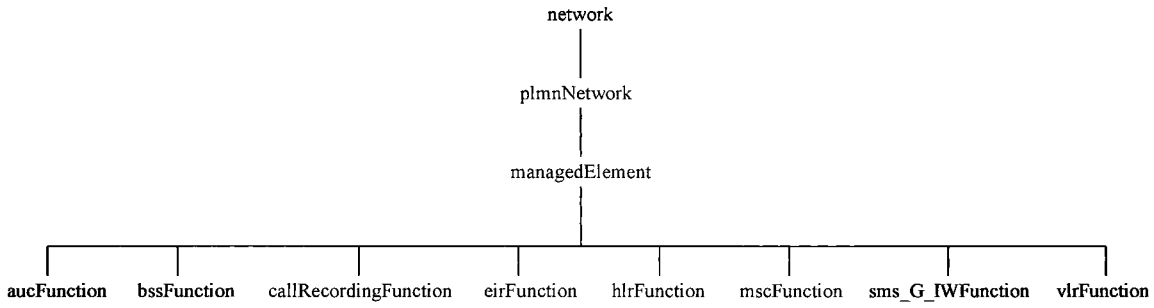


Figure 9-4
Containment hierarchy of GSM object classes.

- *sms_G_IWFunction:* Is used to model the ability of a PLMN to receive and send short messages to a short message service center.
- *vlrFunction:* Is used to model VLR.

9.4.5 Common Management Functions

Some of the management functions are generic in nature and are required by management services. These are known as common management functions. The common management functions treated in Reference 9.1 are forwarding of event reports, logging, and bulk data transfer between an OS and NE.

Managed object classes can emit notifications that are sent in the form of event reports to an OS from NEs. These notifications can be sent to the OS in a controlled manner by using the managed object class eventForwardingDiscriminator (EFD). EFD is defined in X.734 (ISO 10164-5), Event Report Systems Management Function.

Notifications emitted by managed object classes are required to be stored locally in NEs so that they can be accessed later by the OS. The OS retrieves these logged data either in bulk or selectively. The details of logging are defined in X.735 (ISO/IEC 10164-6), Log Control Systems Management Function.

9.4.5.1 FTAM for Bulk Data Transfer. There are cases where a large amount of data has to be transferred between an OS and an NE. When large volumes of data have to be transferred, CMIS M-GET may be one of the methods. Another method of transferring a large amount of data as a file is to use FTAM services. We discuss the use of FTAM for transfer of management information between an OS and an NE in detail, as it is an interesting alternative to CMIP for the transfer of a large amount of data in cases such as performance management and software downloading. Besides, the author has noticed, while implementing network management solutions for wireless communications, that FTAM is very useful in many cases.

As an example, performance on selected attributes can be stored every 15 minutes in files in an NE. There can be a policy to transfer four performance files every hour from NEs to the OS for analysis and reporting. Depending upon the data collected, the sizes of these files can vary from small to very large. FTAM is a suitable means to transfer these files from NEs to the OS. Similarly, when there are cases in which data is not visible at management interfaces, FTAM is a good alternative for data transfer.

GSM 12.00 (Reference 9.1) identifies the following cases of data transfer using FTAM:

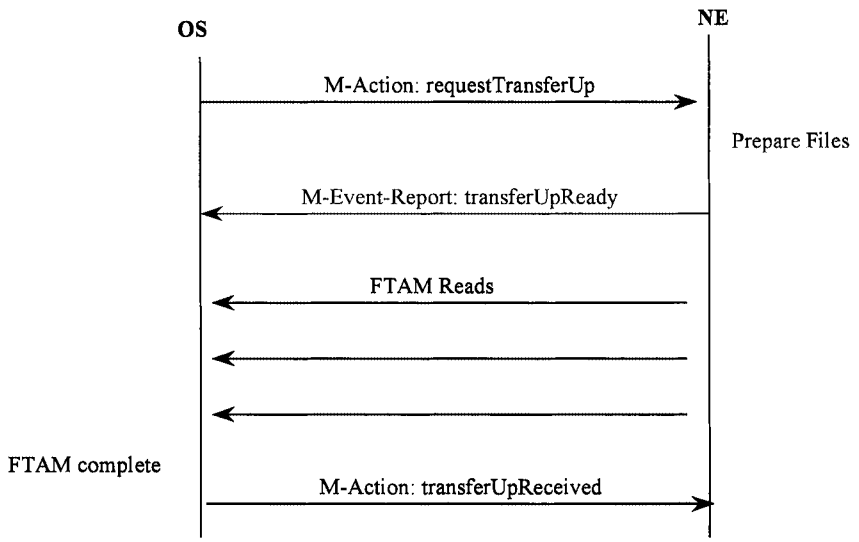
- *Data from NE to OS (requested by OS):* This scenario can be used to perform bulk transfer from NEs to the OS of log files, data files, or object instance data. This case is shown in Figure 9-5.
- *Data from NE to OS (requested by NE):* This case can be useful in performing transfer from NEs to the OS of threshold-related bulk data or bulk data. This case is shown in Figure 9-6.
- *Data from OS to NE (requested by OS):* This is useful when software upgrades or new software must be downloaded to an NE, or multiple managed object commands must be transferred. This case is shown in Figure 9-7.

9.4.6 Different Protocol Layers and Standards

As observed in Chapter 7, CMISE does not include services for the establishment and release of application associations. CMISE services rely on ACSE for the control and release of application associations. In the application layer, normal operation mode is used. For management protocols, we use CMISE.

File transfer capability can be used for software downloading or transfer of data files. File transfer access and management (FTAM) ASE and

Figure 9-5
Data from NE to OS requested by OS.



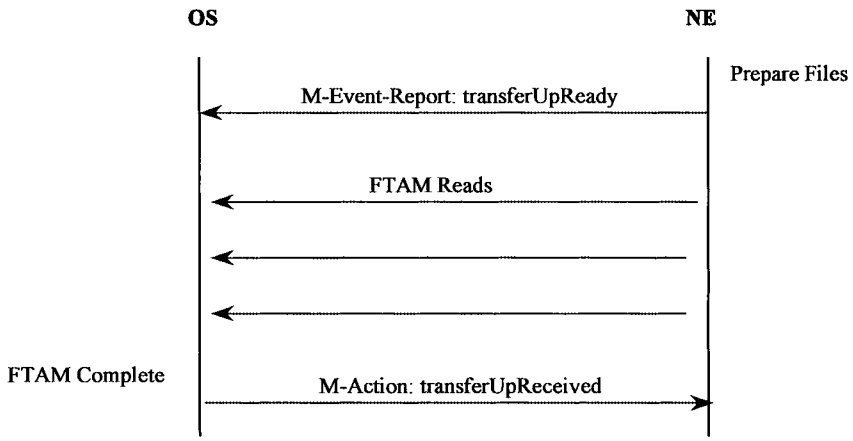


Figure 9-6
Data from NE to OS requested by NE.

ACSE provide this capability. FTAM implementation uses ISO 8571-4, File Transfer, Access and Management—Part 4: File Protocol Specification. FTAM requires kernel, read, and grouping functional units for file transfer between manager and NE.

Each OSI layer has QOS parameters that refer to the characteristics provided by each layer between the endpoints of a connection.

For presentation services, we use presentation protocols defined in X.226 (ISO 8823), Presentation Protocol Specification for Open System Interconnection (OSI) for CCITT Applications. Here the functional units supported are based upon the session layer requirements. However, the kernel presentation functional unit implementation is a basic require-

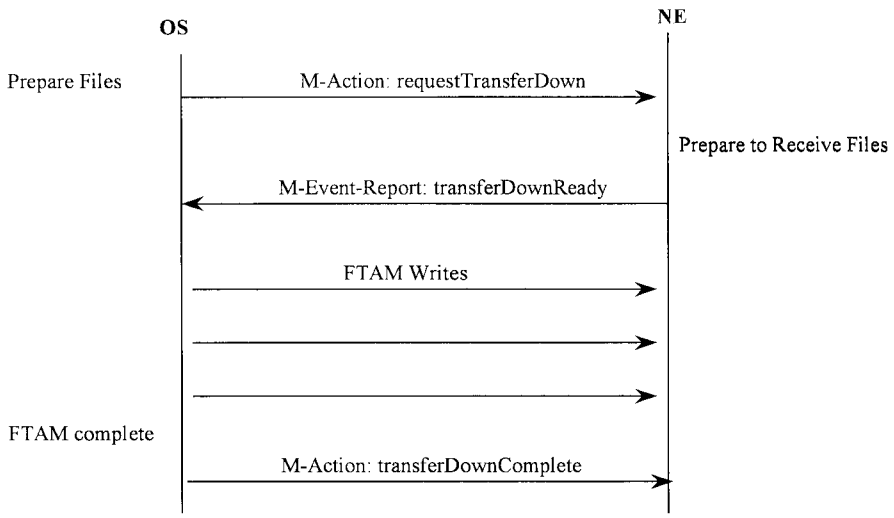


Figure 9-7
Data from OS to NE requested by OS.

ment. ASN.1 (X.208) and BER (X.209) are used for the specification of application protocols and for transfer syntax.

For the next lower layer, the session layer, we use X.225 (ISO 8327), Session Protocol Specification for Open System Interconnection for CCITT Applications. Kernel and duplex are the required functional units. For providing recovery at this layer, the minor synchronize and resynchronize functional units are required. For simultaneous establishment of session, presentation, and application connections, version 2 of session protocol is required.

For the transport layer we use X.224 (ISO 8073), Transport Protocol Specification for Open System Interconnection for CCITT Applications. The class and options used depend upon the user requirements including QOS parameters and the network service provided. The default transport layer settings are:

- Class 2 for SCCP for end-to-end flow control.
- Class 2 for X.25 PSPDNs or ISDNs for multiplexing enabling and flow control (need not be used if the network provides end-to-end flow control).
- Class 0 for X.25 PSPDNs or permanent connection or ISDNs (when multiplexing is not required).
- Class 4 for OSI-LAN.

For the lower layers (1—3), GSM 12.01 (Reference 9.2) uses the examples of SS#7, X.25, ISDN, and OSI-LAN. This scenario is shown in Figure 9-8.

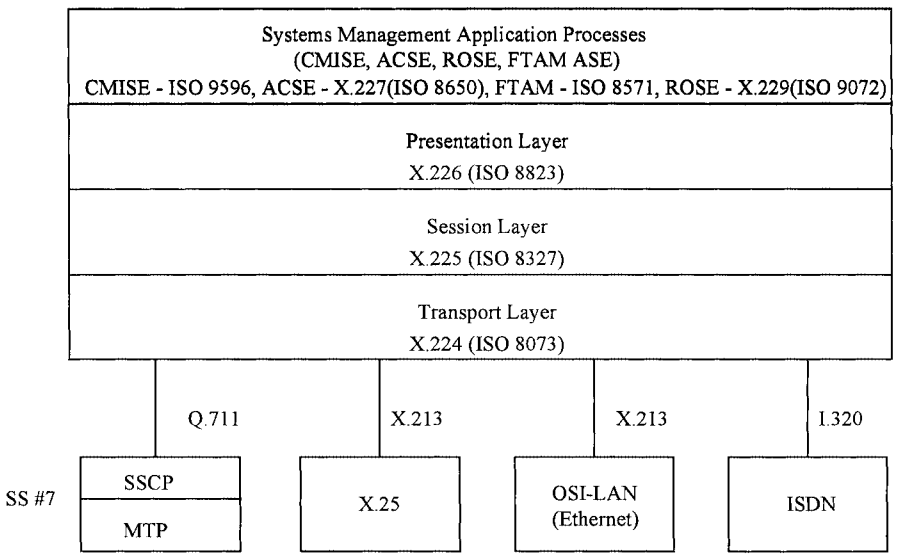


Figure 9-8
Different protocol layers for GSM network management.

9.5 Summary

In this chapter, we have discussed network management for mobile communications, and have presented a brief overview of issues involved in the network management for mobile communications. For mobile communications, ANSI and ETSI GSM standards are widely accepted; for this reason, we have introduced the ANSI network management for PCS with related issues. As a prelude to the discussion of network management for DCS, we have briefly explained GSM architecture, in order to introduce different components of DCS systems. The chapter ends with a discussion of DCS network management details and issues.

9.6 References

9.1. ETSI, GSM 12.00 (ETS 300 612-1), European Digital Cellular Telecommunication System (Phase 2), Network Management (NM): Part 1: Objectives and Structure of Network Management, 1996.

9.2. ETSI, GSM 12.01 (ETS 300 612-2), European Digital Cellular Telecommunication System (Phase 2), Network Management (NM): Part 2: Common Aspects of GSM/DCS 1800 Network Management, 1996.

9.3. ETSI, GSM 12.07 (ETS 300 612-3), European Digital Cellular Telecommunication System (Phase 2), Network Management (NM): Part 3: Operations and Performance Management. This is a draft document at the time of publication.

9.4. ETSI, GSM 12.02 (ETS 300 613), European Digital Cellular Telecommunication System (Phase 2), Subscriber, Mobile Equipment (ME) and Services Data Administration, 1996.

9.5. ETSI, GSM 12.03 (ETS 300 614), European Digital Cellular Telecommunication System (Phase 2), Security Management, 1996.

9.6. ETSI, GSM 12.04 (ETS 300 615), European Digital Cellular Telecommunication System (Phase 2), Performance Data Measurements, 1996.

9.7. ETSI, GSM 12.05 (ETS 300 616) (2nd ed.), European Digital Cellular Telecommunication System (Phase 2), Event and Call Data, 1998.

9.8. ETSI, GSM 12.06 (ETS 300 617), European Digital Cellular Telecommunication System (Phase 2), GSM Network Configuration Management, 1996.

9.9. ETSI, GSM 12.1 V4.2.0 (ETS 301 251), European Digital Cellular Telecommunication System (Phase 2), Fault Management of the Base Station System (BSS). Draft, 1998.

9.10. Cai, J. and D. J. Goodman, "General Packet Radio Service in GSM," *IEEE Communications Magazine*, vol. 35, no. 10, pp. 122—131, 1997.

9.11. ANSI T1-244-1995, American National Standard for Telecommunications—Operations, Administration, and Provisioning (OAM&P) Interface Standards for Personal Communication Services, 1995.

9.12. T1-TR34 *Technical Report: Network Capabilities, Architectures and Interfaces for Personal Communications.*

9.13. ANSI T1-651-1996, American National Standard for Telecommunications—Mobility Management Application Protocol (MMAP), 1996.

9.14. ANSI T1-651a-1996, American National Standard for Telecommunications—Mobility Management Application Protocol (MMAP)—Extensions, 1996.

9.1.5. Yemini, Y. and G. Moss, "Managing Mobile Networks: From Cellular Systems to Satellite Networks," *The Telecommunications Network Management, Technologies and Implementations,* Aidarous, S. and Plevyak, T. (eds.), New York: The Institute of Electrical and Electronics Engineers, pp. 176—196, 1998.

9.7 Further Reading

Garg, V. K., "Management of Personal Communications Services (PCS) Networks," *The Telecommunications Network Management, Technologies and Implementations,* Aidarous, S. and Plevyak, T. (eds.), New York: The Institute of Electrical and Electronics Engineers, pp. 150—175, 1998.

Rauhala, K., *Baseline Text for Wireless ATM Specifications,* Wireless ATM Working Group, 1997.

ITU-T Recommendation M.3020, TMN Interface Methodology, 1995.

This page intentionally left blank.

CHAPTER 10

Broadband Network Management

10.1 Introduction

In this chapter, we briefly introduce the basic concepts involved in broadband communications as a starting point to discuss broadband network management. We primarily discuss network management for SDH and asynchronous transfer mode (ATM). Network management topics include the ITU-T recommendations and the ATM Forum. As Internet MIBs are popular, we also explain IETF SONET/SDH MIB and ATM MIB. As SONET and SDH are very similar, we do not include much discussion on SONET network management.

The topic of broadband network management covers a vast area and includes plenty of published material. Consult the references at the end of the chapter for detailed information.

There are different interpretations of the term *broadband*. Broadband uses the broadband-integrated services digital network (B-ISDN) architecture and integrates SONET/SDH and ATM into the B-ISDN architecture. Broadband networks use fiber coaxial cable and intelligent network elements. In comparison to broadband communications, narrowband communications uses copper wires and carries voice communications. POTS is an example of narrowband communications.

10.2 B-ISDN Protocol Reference Model

As B-ISDN forms the basis for broadband communications, it is necessary to briefly introduce the basic concepts. ITU-T Recommendation I-321 (Reference 10.1) explains the B-ISDN protocol reference model (PRM). B-ISDN architecture (Figure 10-1) forms the basis for technologies used in broadband communications, such as ATM, SDH, and SONET. Different components of B-ISDN protocol reference model are the following:

- *Physical layer:* This layer is dependent upon the physical medium used. An important function of this layer is to synchronize the transmission and reception of continuous information flow in the form of bits.
- *ATM layer:* This layer is independent of the physical layer and is responsible for sending and receiving fixed-size cells between a user node and a network node. As there can be one or more logical connections between two interfaces, this layer handles multiplexing of cells from virtual connections at the originating end point and demulti-

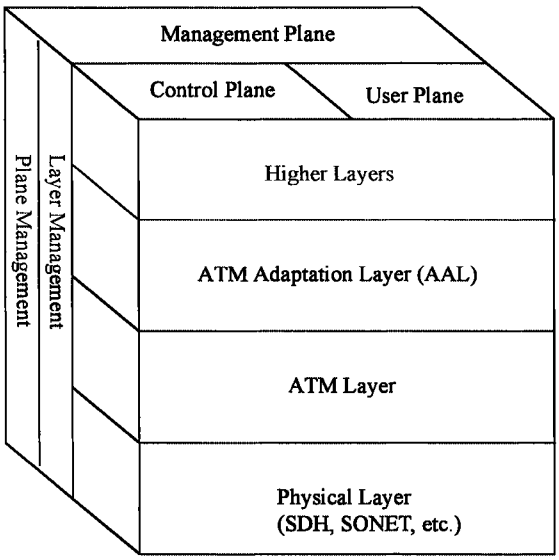


Figure 10-1
B-ISDN protocol reference model.

plexing of cells at the terminating end point. Virtual path identifier (VPI) and virtual channel identifier (VCI) are used to identify a virtual connection. VPI and VCI processing is done in this layer. Flow control information for cell headers is also generated in this layer.

- *ATM adaptation layer (AAL):* This layer primarily transforms higher-layer information to fixed-size ATM cells that are suitable for the ATM layer when sending information. When receiving information, this layer transforms the fixed-size ATM cells to the format of the higher layers.
- *Higher layers:* These layers contain the switched virtual circuit (SVC) signaling protocol, CMIP, SNMP, FTAM protocols, and so on.
- *Control plane:* This plane's functions manage call control, network connection setup, and network connection release. The control plane is required for SVCs and is not required for permanent virtual circuits (PVCs).
- *User plane:* This plane controls user information transfer, including flow control and error recovery.
- *Management plane:* This plane consists of layer management and plane management.
- *Layer management:* This concerns the management of resources and parameters in entities and the operation and maintenance (OAM) operations of each of the ATM protocol reference model layers.

- *Plane management:* This covers the overall management of a system and coordinates management between different planes.

The B-ISDN physical layer may include SONET or SDH. SONET provides protocols for the optical transmission interface. The SONET interface was originally developed by Bellcore and subsequently was standardized by ANSI. SONET hierarchy specifies digital data rates of multiples of 51.84 mbps, with that being the lowest rate. SONET is widely used in North America.

ITU-T has developed synchronous digital hierarchy (SDH) standards that are compatible with SONET for the high-speed transmission of data over optical fiber. The lowest data rate for SDH is 155.52 mbps. SDH is popular in Europe. ITU-T recommends the use of ATM with SDH in the physical layer. SONET can also be used in the physical layer instead of SDH.

We have provided a brief overview of B-ISDN and SDH. There are many good reference books for those who are interested in B-ISDN and SDH (References 10.19, 10.20, and 10.21).

10.3 SDH Network Management

It is very important to know the terms used in SDH before we jump into the discussion of SDH management. This will make it easier for readers to understand and appreciate SDH network management. We therefore briefly introduce the ITU-T Recommendations, which explain the important terminology used in SDH.

ITU-T Recommendation G.805 (Reference 10.5) broadly splits a telecommunications network into a transport functional group and a control functional group. The *transport functional group* transfers telecommunications information from one point to another. The *control functional group* is responsible for performing ancillary services, operations, and maintenance functions.

ITU-T Recommendation G.805 provides the generic functional architecture for transport networks. The functional architecture divides telecommunications networks into functional layers and reference points between layers. The adjacent layers are described in terms of client/server relationships. ITU-T Recommendation G.803 (Reference 10.4) has the specific functional architecture for transport networks based on SDH. Refer to these recommendations for details on the SDH terms used in the following discussion.

ITU-T Recommendation G.831 (Reference 10.6) describes the requirements and processes involved in the management of SDH networks. SDH

management permits the management of transmission networks and the fabric supporting them. A *fabric* is responsible for establishing and releasing cross-connections and is represented as a fabric MOC in M.3100. Some of the important SDH management requirements are the following:

- Set up, maintain at assured performance levels, and continuously monitor paths between all client access points, across all domains, and across network operator boundaries.
- Identify and restore failed branches. Should also be able to inform external network operators or domain management systems about problems.
- Perform simple remote maintenance actions on equipment supported by NEs.
- Identify significant network points, such as access points.
- Set up, validate, and monitor trails. Once a path or section has been set up, it should be validated to determine whether the access points are correct. The path or section should be continuously monitored to ensure that the transmission integrity does not cross a predetermined threshold. If the transmission integrity falls below the threshold, it is treated as a defect. If there is a defect, the path or section has to be restored to the original transmission integrity.

SDH management is described in detail in ITU-T Recommendation G.784 (Reference 10.7). SDH management architecture uses distributed network management principles. Each network management tier provides some network management capabilities. The lowest tier includes SDH NEs, which provide transport services. An NE has an MAF and an MCF. As we have seen in Chapter 2, the MCF is the vehicle for exchanging management information between peers. The MAF can have agent roles only, manager roles only, or both manager and agent roles. Managers in the MAF manage other NEs and communicate with SDH OS/MD. The SDH management hierarchy also includes SDH MD and SDH OS.

For easy management, SDH management is functionally partitioned into the SDH management network (SMN) and the SDH management subnetwork (SMS), as shown in Figure 10-2. SMN includes a subset of TMN functions and is responsible for managing SDH NEs. As can be seen in Figure 10-2, an SMN includes more than one SMS.

For communications between SDH NEs (Figure 10-3), an embedded control channel (ECC) provides a logical operations channel between SDH NEs. The ECC utilizes a data communications channel (DCC) as the physical layer. There are two DCC channels, with speeds of 192 and 576 kbps. SDH NEs can access the lower-speed channel. The high-speed channel

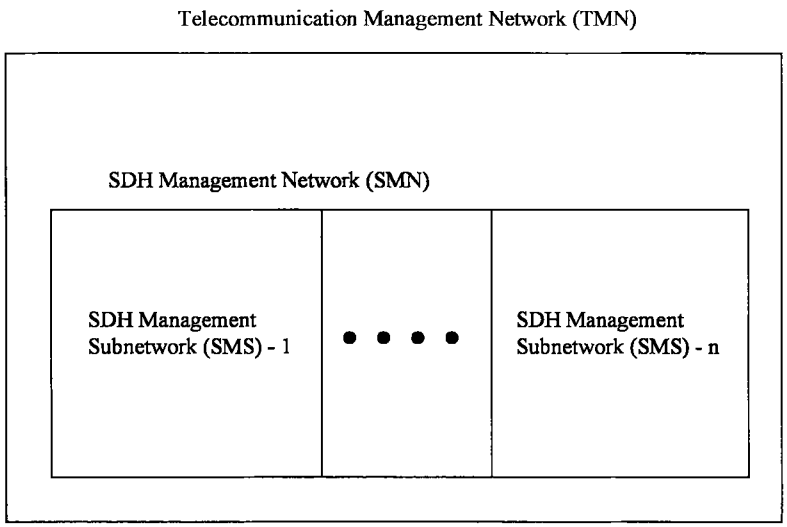


Figure 10-2
Relationship between TMN, SMN, and SMS.

(576 kbps) can be used to support communication between the OS and NEs and between the OS and other OSs.

Each SMS must have at least one gateway network element (GNE) (Figure 10-3) connected to an OS/MD. The GNE acts as an intermediate system, providing network layer routing for ECC messages to end systems in the SMS. Some of the salient points of a SMS are the following:

- A single site may have multiple NEs.
- All NEs must act as end systems.
- Some NEs may be required to perform functions of intermediate systems. This means that ECC messages have to be routed according to the routing control information in the NEs.
- NEs may be required to support Q and F interfaces.
- SDH NEs can communicate using an ECC or via a local communications network (LCN). The LCN is suggested as an alternative to the use of an ECC. The LCN can also be used for communication between SDH NEs and non-SDH NEs (NNEs).

An SDH NE is connected to other components of TMN as shown in Figure 10-3. The different possibilities of NE connections to other components are as follows:

- Workstation using an F interface
- Mediation device using a Q interface,

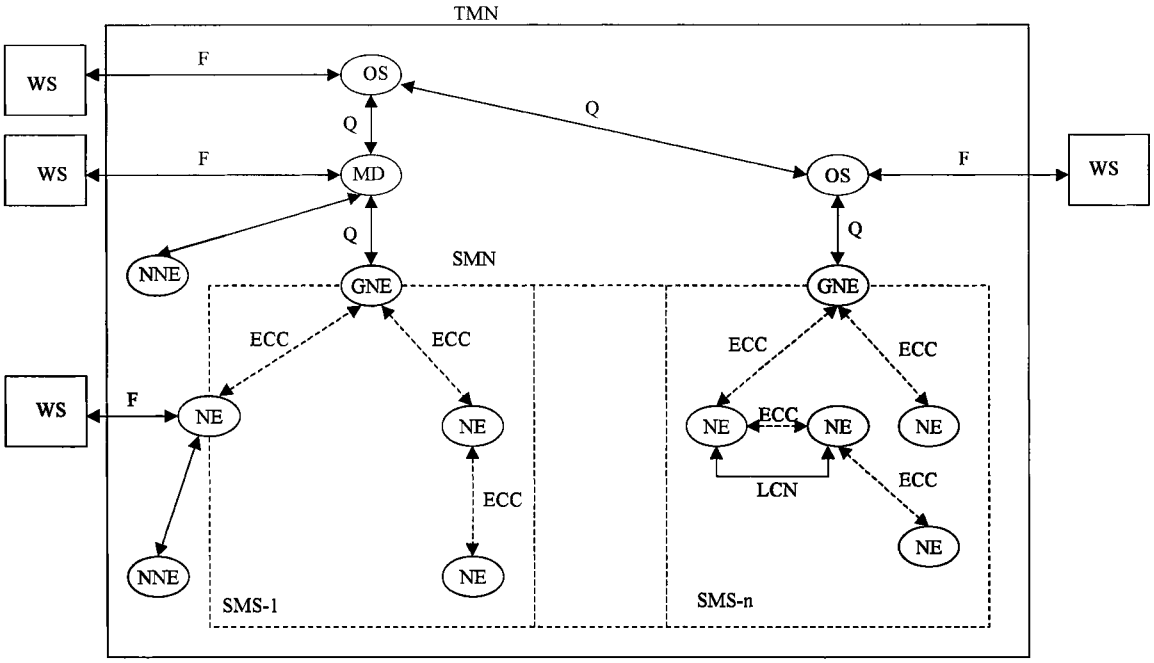


Figure 10-3
SMN, SMS, NE and interfaces.

- Operations system with a Q interface
- Non-SDH NE (NNE) or site-related information

SDH NEs need to manage the ECC. Some of the ECC management functions are as follows:

- Retrieval of network parameters such as packet size, time outs, quality of service, window size, and so on
- Management of network addresses
- Retrieval of operational status of the DCC at a node
- Enabling and disabling of access to a DCC
- Message routing between DCC nodes

10.3.1 Fault Management

Some of the fault management functions, such as testing and external events, have not been defined. However, alarm surveillance and alarm his-

tory management for SDH have been defined. We have already looked into the alarm surveillance function in detail in Chapter 3. The alarm-related functions supported are the following:

- Autonomous reporting of alarms
- Request for reporting of all alarms
- Reporting of all alarms
- Control of alarm reporting, which permits suppression of some alarms and reporting of selected alarms
- Reporting on the status of alarm reporting

ITU-T Recommendation G.784 (Reference 10.7) furnishes details of SDH-specific parameters for which alarm conditions are reported. Alarm history management describes how alarms are stored for future reference. Alarms are recorded in registers with relevant parameters. The OS retrieves these alarms periodically or on demand. The OS controls how the alarms can be stored in registers. A register can either wrap around or stop when the register is full.

10.3.2 Performance Management

Performance management includes performance data collection. Recommendation G.784 includes SDH-specific performance parameters on which data are collected. The strategy of data collection and storage is quite important. Performance data must be available for analysis at a later stage to pinpoint and locate sources of faults.

The performance data is stored in registers in NEs. Each performance event is stored in a register. Note that the registers can be files. The performance data can be collected at 24-h or 15-min intervals. Data collection is done in the current registers. At the end of each data collection period, the data are time stamped and transferred to recent registers. After the transfer of data from the current registers to the recent registers, the current registers are reset to zero.

The 15-minute time interval can be selected when you want to monitor the performance events very closely. Here, at least 16 registers are selected. The current data is transferred to the recent register at the end of each 15-min interval. When the stack of registers is full, the oldest information is discarded. OSs retrieve the performance data at regular time intervals or on demand, based on the need to analyze the performance data.

The OS also controls the threshold settings on SDH parameters. When thresholds are crossed, notifications are sent from the NEs to the OS.

10.3.2.1 Configuration Management. The configuration management for SDH is not fully defined. A very important part of configuration management—namely provisioning—has not been defined. The installation function is also not defined. However, the status and control functions have been defined. The following operations are defined for status and control:

- Operate and release manual protection switching.
- Operate and release force protection switching.
- Operate and release lockout.
- Request and set automatic protection switching (APS) parameters. APS features enable the alternate optical lines to take over when the normal optical lines or interfaces fail.

ITU-T Recommendation I.751 (Reference 10.3) describes the ATM NE management. This recommendation also includes the SDH management functions. The following SDH management functions have been identified:

- *Transport management:* This includes the management of transmission media layers and lower- and higher-path layers. Fault management and configuration management of transport-related layers also have to be done.
- *Connection management:* This covers the configuration and fault management of higher- and lower-order connections and subnetwork connection protections.
- *Synchronization:* This covers the configuration and fault management of timing reference points.
- *Performance management:* This covers monitoring performance parameters such as gauges and counters associated with SDH layer networks and protection switch events.
- *Overhead management:* This includes the fault and configuration management of SDH overhead fields.

10.3.3 SDH Managed Object Classes

ITU-T Recommendation G.774 (Reference 10.8) describes the information model for the SDH NEs. SDH-specific managed object classes are sub-

classed from the MOCs at the termination point fragment defined in M.3100. If the trail termination and adaptation functions are combined in one class, then this combination is represented as a trail termination point class. If the functions are separate, then adaptation functions are represented by a connection termination point class and the trail termination functions are denoted by a trail termination point class (Reference 10.17). For managing connections, the connection point is represented as a cross-connection class. Classes required for testing, filtering, scoping, and logging are used from the MOCs defined in the X.700 series documents.

Refer to Recommendation G.774 for more information on SDH-specific MOCs.

10.3.4 SDH Management Protocol Stack

SDH management information related to OAM&P messages is transmitted using the protocol stack specific to SDH. In the physical layer, SDH DCC is used. For the data link layer, link access protocol on D channel (LAPD) is used. The data link layer should support both the unacknowledged and acknowledged information transfer services. Acknowledged information transfer service is the default mode of operation. The network layer protocol conforms to the network services specification in Q811 for the lower-layer profiles of the Q3 interface. The transport, sessions, and applications layers support the specifications laid out in Q812 for the upper-layer profiles of the Q3 interface. Application layer options such as CMISE, ROSE, and ACSE are also used.

10.4 SONET Network Management

We begin the discussion of SONET network management with a brief overview of SONET. In this section we cover the IETF SONET/SDH MIB. As SONET network management is similar to SDH network management, we cover it only briefly.

10.4.1 Overview of SONET Architecture

SONET architecture has four layers, as shown in Figure 10-4. ATM layers—the virtual path and virtual channel layers—operate on the top of the SONET layers. The SONET layers are the following:

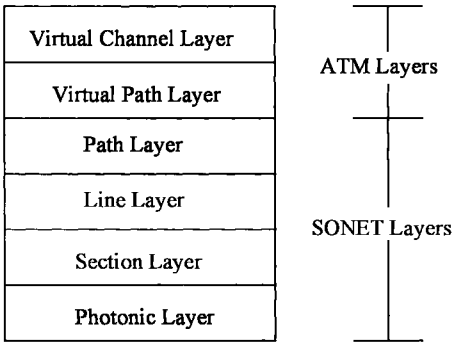


Figure 10-4 Relationship between SONET and ATM layers.

- *Photonic layer:* This is the physical layer. It includes specifications for the optical fiber to be used, details such as the dispersion characteristics of the transmitter, and the sensitivity of the receivers to be used to receive transmitted lasers.
- *Section layer:* This layer creates SONET frames and converts electronic signals to photonic signals.
- *Line layer:* This layer performs functions such as synchronization, multiplexing of data to SONET frames, switching, and so on.
- *Path layer:* This layer performs end-to-end transport of data.

Work on implementing interoperable SONET products and services based on open industry and international standards is currently being done by the SONET Interoperability Forum (SIF). The SIF is a nonprofit consortium of equipment vendors, service providers, and end users. Refer to the SIF Web pages for more details on the SIF documents.

Note that the SONET TMN architecture (Reference 10.16) is only a draft document. File transfer is included in the SONET TMN architecture. FTAM-based file transfer facilitates software downloads and remote backup management. The file transfer function is further broken down to the file transfer initiator function (FTIF) and file transfer responder function (FTRF) blocks. The roles of the FTIF and FTRF are similar to those of the directory server agent (DSA) and directory user agent (DUA). Here, the FTIF and FTRF are part of the OSF and the FTIF is within the NEF. Also, additional reference points have been added to include the FTIF and FTRF function blocks.

10.4.1.1 SONET/SDH MIB. SONET/SDH managed objects are defined in RFC 1595 (Reference 10.11). As can be seen in Figure 10-5, sonetMIB is derived from the transmission object. The primary SONET/SDH objects

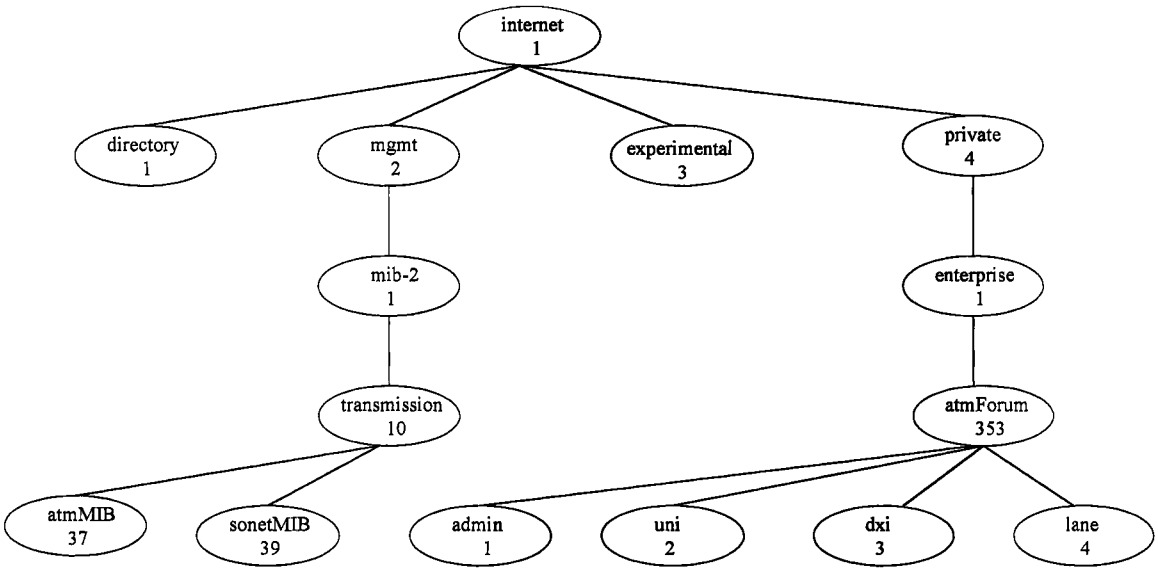


Figure 10-5
SONET/SDH and ATM MIB registration hierarchy.

are sonetObjects, sonetObjectsPath, sonetObjectVT, and sonetConformance (Figure 10-6). The main object groups in the SONET/SDH MIB are the following:

- *sonetMedium:* These objects are used to collect data on the electrical and optical configuration parameters. Data collection is done on sonetMediumLineCoding, sonetMediumLineType, and sonetMediumCircuitIdentifier. Also, counters are maintained on the elapsed seconds and partial seconds since the start of error measurement and on the number of previous intervals for which data is stored. The minimum number of intervals is 4 and the maximum number of intervals is 96. The default value is 32. The time interval is 15 min. Data is collected on the following parameters:
  - *sonetMediumLineCoding:* This includes measurement on the line coding for the SONET/SDH interface. The interface type—whether it is a SONET or SDH interface—is given by sonetMediumType. SONET interface has a value of one and SDH interface has a value of two.
  - *sonetMediumLineType:* This refers to the line type of this interface. The line types are single mode or multimode fiber, or coaxial and UTP for electrical interfaces. Those line types that do not belong to any of these interfaces are included in the sonetOther value.