ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

# INFORMATION INTERCHANGE FOR REMOTE MAINTENANCE AT THE INTERFACE BETWEEN DATA PROCESSING EQUIPMENT AND PRIVATE SWITCHING NETWORKS

## ECMA TR/45

December 1987

Free copies of this document are available from ECMA,
European Computer Manufacturers Association
114 Rue du Rhône – 1204 Geneva (Switzerland)

ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

# INFORMATION INTERCHANGE FOR REMOTE MAINTENANCE AT THE INTERFACE BETWEEN DATA PROCESSING EQUIPMENT AND PRIVATE SWITCHING NETWORKS

## ECMA TR/45

December 1987

## BRIEF HISTORY

This ECMA Technical Report is considered as a precursor to an ECMA Standard within a series of standards for the connection of Data Processing Equipment (DPE) to Private Switching Networks (PSNs).

It represents the status of the technical work performed by ECMA on the advanced definition of a framework and of protocol options for the information interchange for remote maintenance at the DPE-to-PSN interface.

However, this Technical Report is not finalized as a standard due to its companion relationship with currently unstable ISO, CCITT and CEPT documents in the areas of upper layer architecture, common management services and ISDN user-to-network maintenance aspects. It is anticipated that the Technical Report will be used as the main input for the production of a future ECMA Standard after these bodies have made significant progress.

This Technical Report uses abstracts of ECMA TR/34 "Maintenance at the interface between data processing equipment and private switching network". It replaces the version dated October 1984, which no longer matches the directions of the latest studies in ECMA and CCITT.

It is intended that the future standard be based on the ISDN concept as developed by CCITT and also be within the framework of standards for Open Systems Interconnection as defined by ISO. It reflects the practical experience of ECMA Member Companies world-wide, and the results of their active participation in the current work of national and international standardization bodies such as ISO, CCITT and CEPT. It represents a pragmatic and widely based consensus of ECMA Member Companies.

Adopted as an ECMA Technical Report by the General Assembly of ECMA on 10th December, 1987.

# TABLE OF CONTENTS

## 1. SCOPE

This ECMA Technical Report specifies the procedures, the data flows and the message structures for the exchange of information related to the maintenance of the communications aspects of DPE-to-PSN interfaces.

It defines an Application layer protocol implementing the maintenance procedures between two Application processes, which are either located in remote DPEs or on either side of the DPE-to-PSN interface and which are considered as OSI end systems.

It also deals with the mapping of the management application layer process primitives into lower layer services.

The internal structure of the System Management (SM) of OSI end systems is beyond the scope of this Technical Report. Nevertheless, for descriptive purposes, it may be considered that an SM is basically composed of Open System Management Application Processes (OSMAPs), which exchange information related to management tasks, such as:

- configuration management,
- fault management (maintenance),
- accounting management,
- performance management,
- security management.

This Technical Report deals with the exchange of information related to the maintenance of the communications aspects of DPE-to-PSN interfaces. How its maintenance is accomplished is not described by this Technical Report. Normalized mechanisms are provided for the conveyance of information between two open systems for the purpose of maintenance (fault management) for which concepts are defined in ECMA TR/34.

## 2. FIELD OF APPLICATION

This Technical Report applies to the exchange of maintenance information at the PSN's S reference point. The S reference point has been considered, in accordance with CCITT Rec. I.411, as an adequate point for description of the internetworking protocols and for demarcation between the PSN and the terminal suppliers' domains of maintenance responsibility (see Figure 1).

The ECMA ISDN maintenance concepts are defined in ECMA TR/34 and take into consideration basic ISDN features, such as:

- Open communication via the S reference points,
- Portability of DPEs between S reference points, from PSN to PSN, and from ISDN to ISDN.

The ECMA ISDN maintenance concepts are based on defining a user's domain of responsibility for terminal equipment beyond the ISDN's S reference point (as seen from the ISDN or from the PSN).

In general, the ECMA ISDN maintenance concepts define three basic domains of maintenance responsibility:

- a domain for the public network provider (between T reference points),
- a domain for the private network provider (between T and S reference points),
- a domain for the terminal provider (beyond the S reference point).

The ECMA ISDN maintenance concept assumes that every domain provider is capable of determining whether a supposed fault lies in his or in a foreign domain. This should be possible locally and remotely, i.e. across intervening networks, between any maintenance entities.

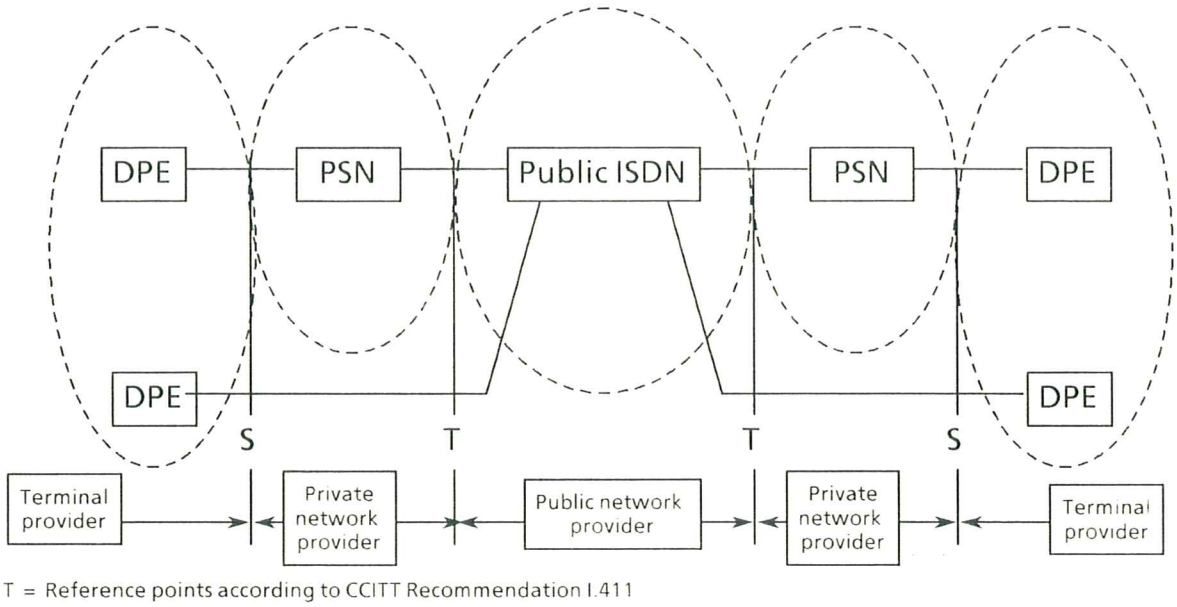



**Figure 1 - ECMA ISDN Maintenance Domains**

## 3. CONFORMANCE

(*Anticipated for a future ECMA Standard:* The minimum requirement to be fulfilled for conformance with the future Standard is to comply with one of the architectural framework options and with the functions necessary to carry out the test call procedure).

A further conformance hierarchy is to be developed.

## 4. REFERENCES


| | |
|---|---|
| ECMA-106 | Layer 3 Protocol for Signalling over the D-Channel at the S-Interfaces between Data Processing Equipment and Private Circuit Switching Networks |
| ECMA TR/24 | Interfaces between Data Processing Equipment and Private Automatic Branch Exchange, Circuit Switching Application |
| ECMA TR/34 | Maintenance at the Interface between Data Processing Equipment and Private Switching Networks |
| ECMA TR/37 | Framework for OSI Management |
| IEEE 802.1 (Draft) | Part B: Systems Management |
| ISO 7498 | Information Processing Systems - Open Systems Interconnection - Basic Reference Model |

| | |
|---|---|
| ISO DP 7498/4 | Information Processing Systems - Open Systems Interconnection - Management Framework (ISO/TC97/SC21/WG4 N93) |
| ISO TR 8509 | Information Processing Systems - Open Systems Interconnection - Service Conventions |
| ISO DP 9595/1 | Information Processing Systems - Open Systems Interconnection - Management Information Service Definition - Part 1: Overview (ISO/TC97/SC21/WG4 N114) |
| ISO DP 9595/2 | Information Processing Systems - Open Systems Interconnection - Management Information Service Definition - Part 2: Common Management Information Service (ISO/TC97/SC21/WG4 N115) |
| ISO DP yyyy/3 | Information Processing Systems - Open Systems Interconnection - Management Information Service Definition - Part 3: Fault Management Information Service (ISO/TC97/SC21/WG4 N116) |
| ISO DP 9596/1 | Information Processing Systems - Open Systems Interconnection - Management Information Protocol Specification - Part 1: Overview (ISO/TC97/SC21/WG4 N121) |
| ISO DP 8649/2 | Information Processing Systems - Open Systems Interconnection - Service Definition for Common Application Service Elements - Part 2: Association Control |
| ISO DP 8822 | Information Processing Systems - Open Systems Interconnection - The Presentation Service Definition |
| ISO DIS 8824 | Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1) |
| ISO DIS 8825 | Information Processing Systems - Open Systems Interconnection - Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) |
| ISO DIS 9072/1[X.ROS0] | Information Processing Systems - Text Communication Remote Operations [part 1]: Model and Service Definitions |
| ISO DIS 9072/2[X.ROS1] | Information Processing Systems - Text Communication Remote Operations [part 2]: Protocol Specification |
| CCITT Rec. I.411 | ISDN User-Network Interfaces - Reference Configurations |
| CCITT Rec. M.2x | Principles for Telecommunication Management Networks |
| CCITT Rec. I.60x | Maintenance of the ISDN Subscriber Access |
| CCITT Rec. X.400 | Message Handling Systems: System Model-Service Elements |
| CCITT Rec. X.409 | Message Handling Systems: Presentation Transfer Syntax and Notation |
| CCITT Rec. X.410 | Message Handling Systems: Remote Operations and Reliable Transfer Service |

| | |
|---|---|
| CCITT Rec. Q.940 | Draft Recommendation for ISDN User-Network Interface Management - General Aspects |
| CCITT Rec. Q.941 | Draft Recommendation for ISDN User-Network Interface Management - Protocol and Service Specification |
| CCITT Rec. Q.942 | Draft Recommendation for ISDN User-Network Interface Management - Management Protocol Applications |

## 5. DEFINITIONS

### 5.1 Alarm

A signal indicating a request for maintenance intervention. It is generated by a System Management Application as a consequence of an event in the system when the system has been disturbed or is in imminent danger of being disturbed so that its total performance is impossible or in danger.

### 5.2 Application Association

Within the scope of this Technical Report, an interconnection of two cooperating application layer entities for the purpose of maintenance transaction exchange. A specific application association is identified by an application association identifier.

### 5.3 Compatible Equipment Selection Parameter (CESP)

The Compatible Equipment Selection Parameter may be assigned at the time of installation and is unique across the user domain. It may by itself be used to uniquely identify a particular terminal.

### 5.4 Data Processing Equipment (DPE)

Specific type of terminal equipment, exclusively or mainly used to process data (in contrast to a voice-only terminal).

### 5.5 Error

A malfunction in the operation of a system.

### 5.6 Event

The occurrence of a significant normal or abnormal condition.

### 5.7 Executor

A system in charge of accomplishing the action required by the requestor.

### 5.8 Failure

A failure may be caused by a fault depending on its severeness and the (non-)existence of redundancy.

### 5.9 Fault

The mechanical or algorithmic cause of a malfunction.

### 5.10 Maintenance

Throughout this Technical Report the term maintenance is used as a synonym for fault management. It does not cover the full range of procedures that are normally covered by this term, such as recovery procedures, repair, preventive maintenance, human intervention.

### 5.11 Maintenance Actions

#### 5.11.1 Maintenance Process

A sequence of one or more maintenance transactions to achieve a maintenance objective. A specific maintenance process is identified by a test number (Test Id).

#### 5.11.2 Maintenance Transaction

The execution of a single step of the maintenance process. A maintenance transaction consists of one or more maintenance operations.

#### 5.11.3 Maintenance Operation

The basic component of a maintenance transaction. Ultimately it will be mapped to a ROS operation.

### 5.12 Private Switching Network (PSN)

A PSN provides switching functions (circuit and/or packet switching). It is operated by the user and located on his premises to cover the communications needs in his domain. Terminal equipment is connected to a PSN at its S reference points.

The term Private Switching Network includes both the private circuit switching network and the private packet switching network.

### 5.13 PSN Termination (PT)

The termination of a PSN at the S reference point.

### 5.14 Reporting System

A system which initiates an unsolicited report of an event.

### 5.15 Resource

An object that can be identified as the target of a maintenance operation. A specific resource is identified by its resource identifier.

### 5.16 Requestor

The system which initiates and, in principle, controls a maintenance transaction.

### 5.17 S Reference Point

In the ISDN user-to-network reference configuration, the S reference point is defined between the DPE and the PSN termination (see CCITT Rec. I.411). ECMA has identified two interfaces at the S reference point, the $S_0$ interface which provides 2 B- and 1 D-channel (basic access), and the $S_2$ interface which provides 30 B- and 1 D-channel (primary rate access).

### 5.18 Significance

A definition or a procedure may have different ranges of application. Internal significance applies to the internal structure of an entity (DPE, PSN) and is not subject to standardization. Local significance applies to peer entities on each side of an interface. Global significance applies to end-to-end configurations.

### 5.19 Terminal Equipment (TE)

Any terminal (voice or data processing or combination of both) connected to a PSN at an S reference point.

## 6. ARCHITECTURAL FRAMEWORK

### 6.1 Overview of the Management Protocol

Within a DPE, system management functions are controlled and performed by the System Management (SM). This entity receives, analyses and makes decisions out of the information generated locally in the Layer Management Entities (LMEs) or received from a remote SM.

The SM, whose description is beyond the scope of this Technical Report, can be seen as a set of Open System Management Application Processes (OSMAPs), communicating with remote OSMAPs by use of one or more Open System Management Application Entities (OSMAEs).

In this Technical Report, the Interface between the OSMAE and the OSMAP is referred to as the System Management Interface (SMI) and the Interface between the OSMAE and the lower layer entity is referred to as the System Management Data Service Interface (SMDSI). An Open System Management Application Process (OSMAP) will communicate with a remote OSMAP by use of an Open System Management Application Entity (OSMAE). See Figure 2.

The layer-specific management functions are provided for each layer by the respective Layer Management Entity (LME). The LME is a component of the layer which effects control of the communication functions of the layer provided by the Protocol Entity (PE). An LME is logically interfaced to the System Management by its Layer Management Interface (LMI).

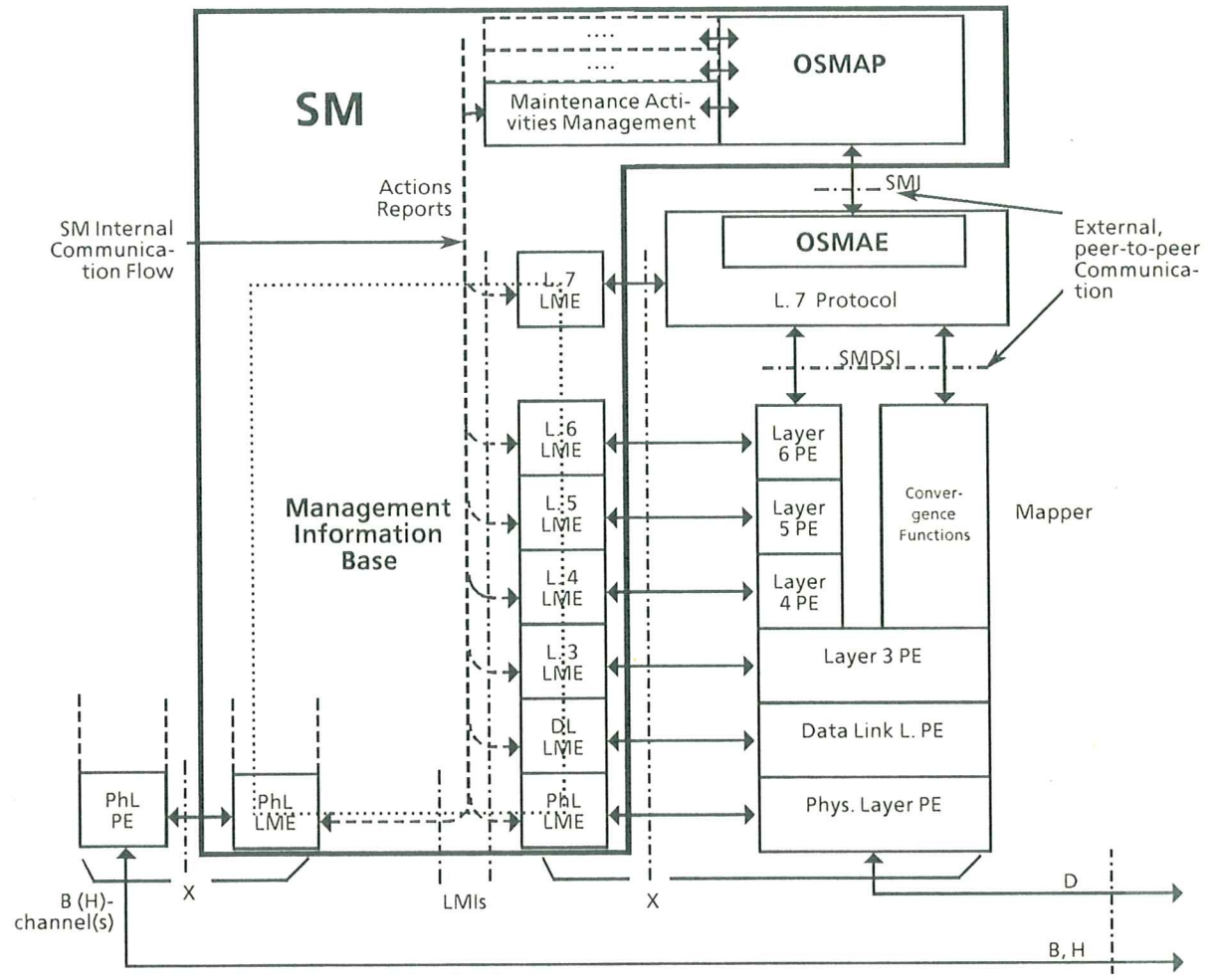
*Note 1*

*In the context of describing the services at the LMI and SMI, the term "interface" is not used in the OSI sense. The LMI and SMI are to be understood as conceptual reference points where maintenance services are provided by the SM.*

Whether or not Layers 4 to 6 are used in an ISDN environment, depends on the actual implementation. A convergence protocol, mapping from the SMDSI directly to Layer 3, could be used instead.

The protocol specified in this Technical Report is a particular example of an OSMAE that provides communication facilities between a *requesting* OSMAP in one end system and an *executing* OSMAP in a second end system. The protocol carries out maintenance activities on behalf of the requesting OSMAP.

The application layer entities which are used for communication between the OSMAPs are called Open System Management Application Entities (OSMAEs). The structure of an OSMAE, its relationship with adjacent entities and the communication rules between peer OSMAEs are defined hereafter.




X = D, B or H-channel protocol stack with its layer internal interfaces, included in layer specifications.

**Figure 2 - Management Architecture for Maintenance Application**

### 6.2 Architectural Concepts and Terminology

The OSMAE requests services from the lower layer and provides services to the OSMAP via service primitives. The primitives represent, in an abstract way, the logical exchange of information and control between adjacent layers. They do not specify or constrain the implementation of entities or interfaces.

The primitives that are exchanged between adjacent layers are of the following four types:

- request,
- indication,
- response, and
- confirm.

The REQUEST primitive type is used when a higher layer requests a service from the next lower layer.

The INDICATION primitive type is used by a layer providing a service to notify the next higher layer of activities related to the REQUEST primitive.

The RESPONSE primitive type is used by a layer to acknowledge receipt, from a lower layer, of the INDICATION primitive.

The CONFIRM primitive type is used by the layer providing the requested service to confirm that the activity has been completed.

Information between peer entities is transferred by means of protocol data units (PDUs).

### 6.3 Open System Management Application Entity (OSMAE)

As shown in Figure 2, the OSMAE is interfaced:

- to the communication channel via the System Management Data Service Interface (SMDSI), and

- to the System Management Entity via the System Management Interface (SMI).

The OSMAE is a Layer 7 entity. It uses the common services defined for the establishment and release of Layer 7 associations (ACSE) and for remote operations (ROSE). The part specific to management is referred to as the Common Management Information Services. Figure 3 depicts the Layer 7 sublayering.

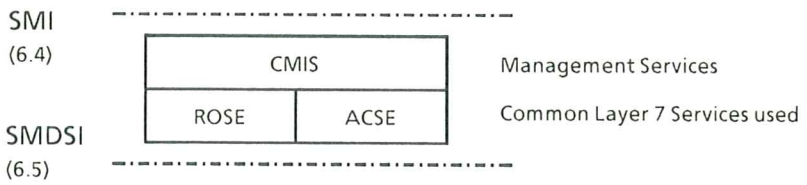



**Figure 3 - Internal Architecture of the OSMAE**

### 6.4 System Management Interface (SMI)

The SMI is the service interface between the System Management and the OSMAE. The two entities exchange M-primitives in order to allow remote maintenance operations. The services provided to the Management entity by the OSMAE are described hereafter. The associated service elements (primitives) REQUEST, CONFIRM, INDICATION and RESPONSE are available for each service. Service element parameters are defined in 8.1.

As a summary the service elements (primitives) required to provide the services at the SMI interface are listed in Table 1.

| Service Element | Type |
|---|---|
| M-INITIALIZE | confirmed |
| M-TERMINATE | confirmed |
| M-ABORT | non-confirmed |
| M-CONFIRMED-EVENT-REPORT | confirmed |
| M-EVENT-REPORT | non-confirmed |
| M-GET-ATTRIBUTES | confirmed |
| M-SET-CHARACTERISTICS | confirmed |
| M-COMPARE | confirmed |
| M-ACTION | confirmed |
| M-BLOCKING | confirmed |

**Table 1 - Common Management Information Service (CMIS) Elements**

*Note 2*

*If the originating OSMAP chooses to use the CL mode of operation to perform management activity, the M-INIT and M-TERMINATE / M-ABORT primitives are not used. In this case, the calling and called OSMAP addresses are presented in each M-operation service element.*

*In both cases, the originating OSMAP uses an association reference in the M-operation primitives, corresponding either to a pre-arranged association between application entities (CL mode of operation) or to a dynamically established association (CO mode of operation).*

### 6.4.1 Initializing Association (M-INITIALIZE)

This facility provides a requesting OSMAP in one system with the capability to start a connection oriented relation by requesting the OSMAE to establish an association between the two application entities that will be involved.

### 6.4.2 Releasing Association (M-TERMINATE / M-ABORT)

This facility provides a requesting OSMAP in one system with the capability to finish a connection oriented relation by requesting the OSMAE to release the previously established association between the two application entities. Different services will be offered dependent upon whether the release may be performed in an orderly manner ("terminate") or abruptly ("abort").

### 6.4.3 Action (M-ACTION)

This facility provides a requesting OSMAP in one system with the capability to request the executing OSMAP in another system to perform actions. Examples of these actions are loopback control and self-test of the interface.

### 6.4.4 Set Attributes (M-SET-ATTRIBUTES)

This facility provides a requesting OSMAP in one system with the capability to request its executing OSMAP in another system to set attributes, e.g. set or change values of parameters, counters, etc. in the executing OSMAP.

### 6.4.5 Get Attributes (M-GET-ATTRIBUTES)

This facility provides a requesting OSMAP in one system with the capability to request its executing OSMAP in another system to read one or more management attributes in the executing OSMAP.

### 6.4.6 Event Report (M-EVENT-REPORT / M-CONFIRMED-EVENT-REPORT)

This facility provides an OSMAP in one system with the capability to transmit unsolicited event information to another OSMAP. This service is an optionally confirmed service, i.e. the RESPONSE and CONFIRM service elements may be issued depending on whether the acknowledged service is requested by the reporting OSMAP.

An event report may be:

- unsolicited, reporting on an event being monitored by the system,

- solicited, reporting the outcome of a previously initiated maintenance process.

### 6.4.7 Compare Attributes (M-COMPARE)

This facility provides a requesting OSMAP in one system with the capability to ask its executing peer to compare certain attributes with a set of associated values. On completion, the executing OSMAP returns the result of the comparison.

### 6.4.8 Blocking Functions (M-BLOCKING)

This facility provides a requesting OSMAP in one system with the capability to request the application in the executing OSMAP to perform multiples of the ACTION, GET, COMPARE and SET functions as a group. This allows the synchronization of operations at the executing side and further requests will not be processed until the completion of the blocked operations.

## 6.5 System Management Data Service Interface (SMDSI)

The System Management Data Service Interface (SMDSI) is the service interface for the exchange of primitives between the lower layer protocol entity and the OSMAE.

The lower layer services typically provide connection establishment and release as well as data transfer in the connectionless and in the connection-oriented mode. They are summarized in Table 2.

Depending on the implementation, these services may be offered by the Presentation Layer (P-type primitives) or be mapped into lower layer services by convergence functions as shown in Figure 2.

| Service | Request | Response | Confirmation | Indication |
|---|---|---|---|---|
| CONNECT | x | x | x | x |
| DISCONNECT | x | x | x | x |
| DATA | x | | | x |
| UNIT-DATA | x | | | x |
| ABORT | | | | x |
| ERROR | | | | x |

**Table 2 - Services Offered at the SMDSI**

### 6.6 Entity Model

The maintenance activities are described as asymmetric functions using symmetrical communication paths in the sense that a maintenance activity is always started by a REQUESTOR or INITIATOR who is asking from an EXECUTOR or AGENT to manipulate objects. These objects may be events or attributes. These can be classified as belonging to individual manageable ENTITIES or resources. Each elementary operation that will have to access or refer to objects will define these objects by specifying first the manageable entity to which it belongs and then identifying the object within the entity.

A hierarchical entity model is defined to allow access to any individual object in a simple way. When a given entity may be duplicated, an entity instance identifier will help to resolve the ambiguity.

The entity model for a PSN interface is as shown on the hierarchical tree presented in Figure 4.

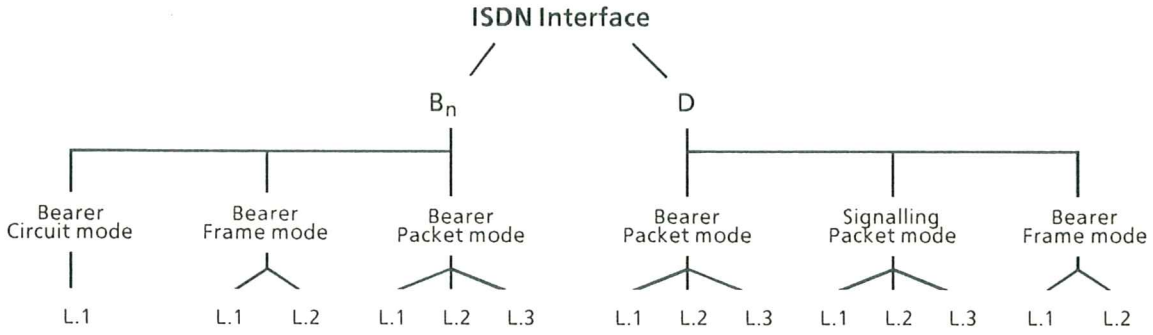



**Figure 4 - Hierarchical Tree for Resource Definition**

The various levels identified are as follows:

- ISDN interface (identifying one access),
  - Channel type and number (B ,D, ...),
    - Information type (signalling, bearer, ...),
      - OSI layer (1, 2, 3).

The attributes and events pertaining to each entity can then be defined implicitly within the entity. Some entities may be empty or partly empty when no object is identified within them. In this case they exist only as a hierarchical level.

The PSN interface entity model only contains entities belonging to the network access functions, i.e. which are involved in the provision of the required bearer service (signalling and lower layer protocols on the bearer channels). The protocols not terminated in the PSN are excluded from the model since they belong to the application part.

The most comprehensive entity is the ISDN interface entity. It corresponds to one access and embraces the D-channel stack (Layers 1 to 3) and, if applicable, the layers of the B-channel involved in the bearer service. Other higher layers of the B-channels are excluded; they are considered as part of the applications. Figure 5 gives an idea of the entity involved in this model. For each entity a set of events and attributes may be defined independently.

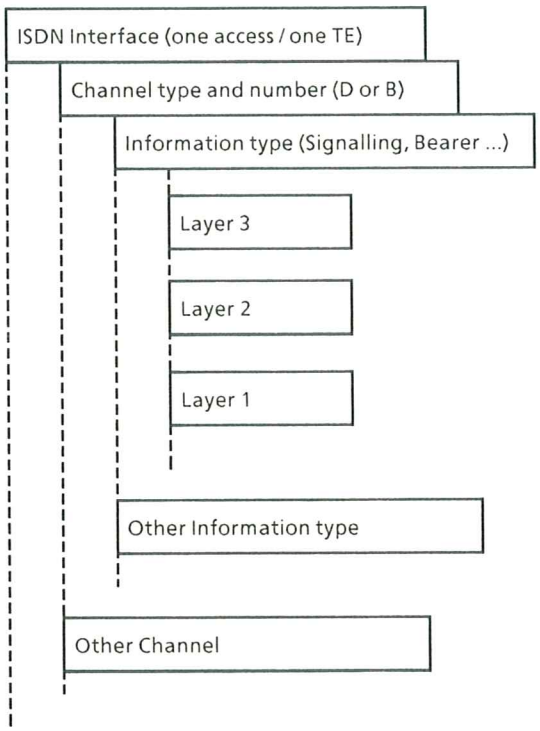



**Figure 5 - Entity Model for the ISDN Access**

*Note 3*

*The identity of the entity at the executing end may not be known to the initiating OSMAP when it requests a maintenance action at the remote end of a connection. In this case the remote Executor will be able to identify the entity by the context of the connection path used to convey the maintenance request.*

*As an example, remote maintenance may be required on an existing B-channel connection. The channel identity is only locally significant at each end. The channel will be inherently the same as that used to convey the maintenance request.*

## 7. GENERAL MAINTENANCE PROCESSES AND DATA FLOWS

This Clause describes in general terms the maintenance procedures for which mechanisms are provided in Clause 8. It describes the exchange of information between the end system that originates a request (known as the REQUESTOR) and the remote system which is asked to execute the particular request (known as the EXECUTOR).

In the case of event reporting, the concepts of REQUESTOR and EXECUTOR are replaced by ORIGINATING (reporting) and TERMINATING entities. The flows presented give an overview of the following maintenance activities:

- Loopback activation/deactivation,
- Self-test,
- Remote monitoring of resources,
- Remote control of resources,

- Event reporting,

- Test calls.

It is assumed that any OSMAP will protect itself against erroneous conditions by means of supervisory timers and counters. The definition and use of these timers and counters as well as subsequent actions on timer expiry are beyond the scope of this Technical Report.

In the following flows only those timers are shown which are considered helpful for clarification.

The various Figures in this Clause indicate the roles (REQUESTOR and EXECUTOR), the type of function that is required and carried by an M-type primitive as per 6.4. An event report is carried by an INVOKE PDU and may be linked to a previous operation. See Clause 8.

## 7.1 Loopback Activation/Deactivation

These procedures belong to the ACTION generic family as described in 6.4.3.

### 7.1.1 Protocol Flow

In order to completely control a loopback test, only the loopback requestor should be allowed to cancel a loopback test. A timer indication will be associated with the request and will run at the executor's side during the test. If the requestor has not required the cancellation of the loopback by time-out as shown in Figure 6, the executor will cancel the loopback on his own, as shown in Figure 7 (this will be the case particularly when the interface is no longer accessible by the requestor).

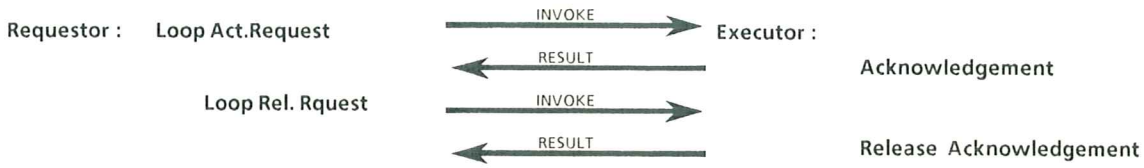



**Figure 6 - Loop Activation / Cancellation by the Requestor**


**Figure 7 - "Blind" Loop Cancellation by the Executor**

It may happen that the executor wants to terminate the loopback for its own reasons. It will send a request for release by the requestor and start a timer, as shown in Figure 8.

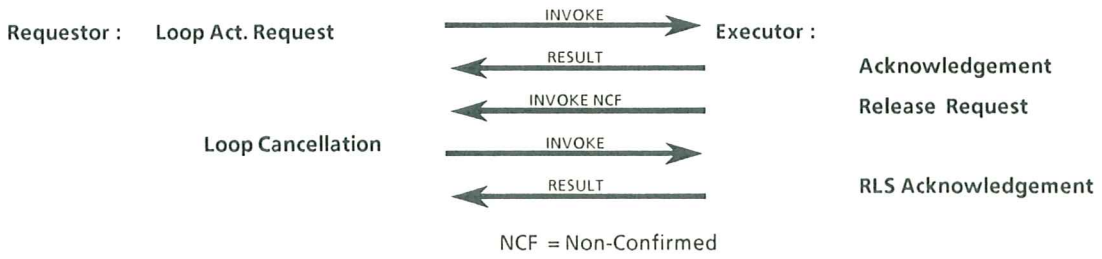



**Figure 8 - Loop Cancellation on Request of the Executor**

If the request for release is not satisfied within the timed period, the executor will terminate the loopback as indicated in Figure 9.


**Figure 9 - Loop Release after Unsuccessful Request for Release**

### 7.1.2 Loopback Test Phase

The requestor will consider the loopback as active on receipt of an acknowledgement.

At the requestor side, the loopback will be considered as unreliable on transmission of the release request or on receipt of a release acknowledgement when not preceded by any other message.

## 7.2 Self-Test Procedures

The procedures described belong to the ACTION generic family as described in 6.4.3.

### 7.2.1 Protocol Flow

A system which does not support self-tests must be able to recognize and reject a self-test request. The self-test activation procedures consist of four phases of information exchange:

i) test REQUEST from the requestor;

ii) test INFORMATION from the executor, in order to inform the requesting end of the test about consequences such as time duration;

iii) test ACTIVATION from the requestor (or request CANCEL); in the light of the information received the requestor can then decide whether or not the test should be activated;

iv) test activation (or cancellation) ACKNOWLEDGEMENT from the executor.

Figure 10 depicts the acceptance of a self-test request.

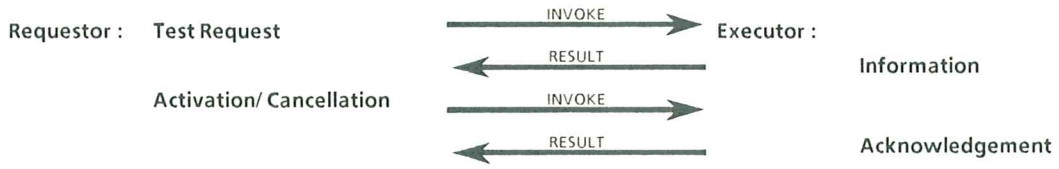



**Figure 10 - Self-Test Activation / Cancellation Accepted**

Figure 11 shows the rejection of a self-test request. The rejection will include an indication of the cause for the refusal, e.g. facility not supported, request not authorized (after a compatibility check), DPE busy in a call, DPE busy in a loopback test.


**Figure 11 - Self-Test Activation / Cancellation Rejected**

The mechanism depicted in Figure 12 allows the executor to return test status information whenever necessary, in order to enable subdivision of self-tests into several phases.


**Figure 12 - Subdivision of a Test into Phases, under the Executor's Control**

When the decision to cancel or to continue the self-test is to be made by the requestor, the executor will signify the end of a phase as depicted in Figure 13.

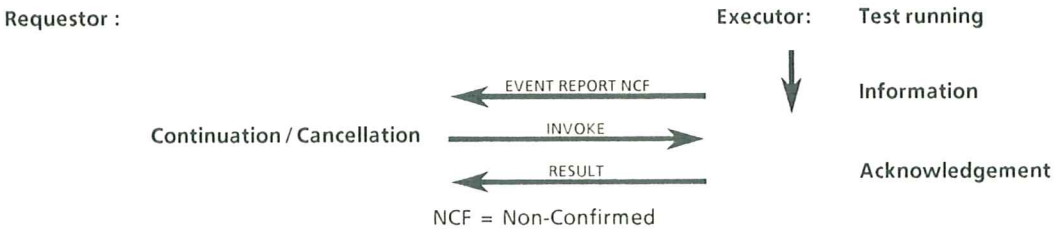



**Figure 13 - Subdivision of a Test into Phases, under the Requestor's Control**

### 7.2.2 Self-Test Phase

Partitioning of a self-test into phases, when implemented, is the responsibility of the OSMAP and is beyond the scope of this Technical Report.

## 7.3 Resources Monitoring / Status Enquiry

Resources monitoring is part of the GET generic family such as described in 6.4.5. Its purpose is to retrieve information on demand from a remote System Management.

### 7.3.1 Protocol Flow

At any time an entity can request its peer to return information concerning the support or non-support of maintenance facilities as well as the interface state (off-line for maintenance, awaiting answer, ...), when a previous request (loopback test, self-test, ...) has been made. This request will also allow statistical information to be retrieved. The requesting end will send a status enquiry request to the terminating end which will return the ad-hoc information or reject the request if it is not admissible. See Figure 14.


**Figure 14 - Reject of a Status Enquiry**

## 7.4 Remote Control of Status / Counters / Thresholds

The remote resources control is part of the SET generic family, as described in 6.4.4. It is aimed at changing information on demand in a remote System Management. The request may have a global or atomistic significance in the sense that the total request may fail if at least one change fails or may be valid as long as there is at least one change accepted.

### 7.4.1 Protocol Flow

The function of SET REQUEST is to request the remote OSMAP to set the specified data elements and to return the status of each data element set to the local OSMAP.

A SET RESPONSE is expected for each SET REQUEST, as shown in Figure 15.

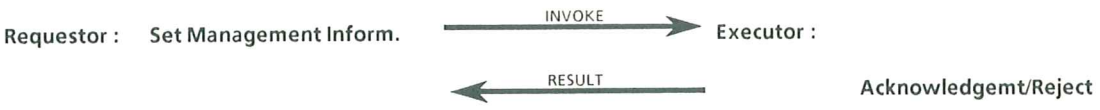



**Figure 15 - Status Control Request**

### 7.4.2 Changing Remote Parameters

When an OSMAP receives an APDU containing a SET REQUEST, it checks the Resource Identifier to determine in which layer(s) the data elements are to be set. Subsequently, the Access

Control field will be checked. In the Data Element List each data element identified is set to the value specified for that data element. If the data element identified is not a valid data element for the SET REQUEST operation, the reason for the failure of setting it will be returned in the status field of the SET RESPONSE APDU.

An OSMAP generates a RESPONSE APDU containing a SET RESPONSE to respond to a SET REQUEST. The information in the Resource Identifier field is the same as that in the corresponding SET REQUEST. The SET RESPONSE may contain either a single status value indicating an invalid request (unsupported operation, bad layer internal selector, no access authentication, etc.) or a Data Element List containing the value appropriate for the corresponding request.

## 7.5 Event Reporting

This function is to notify the remote OSMAP of any event that has been detected by the reporting OSMAP. It is part of the M-REPORT family described in 6.4.6.

An event acknowledgement is not expected for each event reported, but may be requested in the EVENT REPORT.

The function of an event acknowledgement is to acknowledge the receipt of an EVENT REPORT by an OSMAP. The OSMAP reporting the event has the option of requesting that its EVENT REPORT be acknowledged.

### 7.5.1 Protocol Flow

The event message is sent by a reporting OSMAP whenever a particular change of a state occurs that has been appointed as an event to be reported remotely. The receipt of an EVENT REPORT is unsolicited in the sense that the OSMAP reported to did not make specific requests for events.

The information provided by the layers (LMEs), the request for acknowledgement and the type of event are supplied in appropriate fields. Multiple events may be associated in a single REPORT. The inclusion of the event's time is optional.

The OSMAP reported to will not return any acknowledgement APDU unless it is asked to do so. In this case an event acknowledgement is returned at the earliest opportunity in order to cancel the timer that may have been started by the OSMAP reporting the event. This is shown in Figure 16.

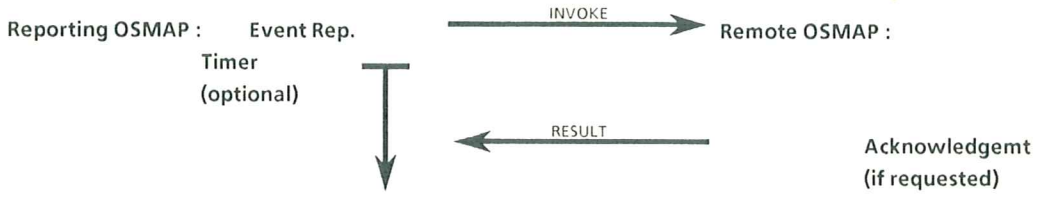



**Figure 16 - Event Reporting**

## 7.6 Test Call

The procedures described hereafter belong to the ACTION generic family as described in 6.4.3. A test call is a simple action with immediate acknowledgement from the executor. It does not require the use of a bearer channel and will be carried out by a simple exchange of information. If it is

successful, it will prove the good functioning of the end-to-end communication channel up to the OSMAP level as depicted in Figure 17.

Failures will be detected by the use of supervisory timers of the requesting OSMAP.

### 7.6.1 Protocol Flow

The test call procedure is mandatory. Its protocol flow is shown in Figure 17.


**Figure 17 - Test Call (Successful)**

### 7.6.2 Test Call Phase

The test call phase consists of an immediate response reaction of the executing OSMAP to the test call indication and there is no partitioning of a test call into different phases.

## 7.7 Access Control

The purpose of this function is to prevent a remote OSMAP against unauthorized maintenance transactions. This function may be used in conjunction with any of the previously described maintenance processes.

Access control may be executed during the association establishment phase and/or for individual operations. Access control information is carried as a parameter in the ACSE establishment message or as an argument in the APDU. The structure of this parameter or argument is beyond the scope of this Technical Report.

When the receiving OSMAP does not accept the access control information offered by the requesting OSMAP, it may return an error with the appropriate cause. This is shown in Figure 18.

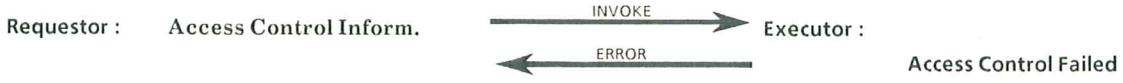



**Figure 18 - Failure of Access Control**

If the Access Control field is formed incorrectly an error returned will be sent back to the requesting OSMAP. Optionally the executing OSMAP may decide to not respond at all.

# 8. OPERATIONAL PROCEDURES

This Clause describes the primitives and their procedures at the SMI and at the SMDSI. Furthermore, the protocol between two peer OSMAEs is described. The partitioning into Clauses 8 and 9 is shown in Figure 19.

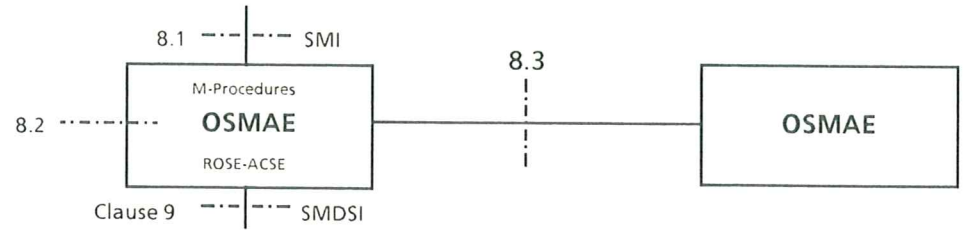



**Figure 19 - Structure of Clauses 8 and 9**

## 8.1 Primitives and Primitive Procedures at the SMI

The SMI primitives are passed from the requesting OSMAP to the CMIS with additional detailed information contained in the primitive parameters.

The OSMAE procedures resulting from the reception of the SMI primitives presented in Table 1 are presented hereafter.

Depending on the activity undertaken by the OSMAP a connection-oriented (CO) mode or connectionless (CL) mode can be used.

When a CO mode is desired by the requesting OSMAP, the M-INITIALIZE and M-TERMINATE / M-ABORT are required to establish or release the context which will be maintained throughout the particular activity. The established context will be referred to by the OSMAP with an association reference parameter to be internally generated by the ACSE and transferred to the OSMAP via the M-INITIALIZE indication and confirmation elements.

The association reference has only internal (i.e. local) significance; it is not exchanged in the context establishment APDU, it is present in all the M-operation primitives belonging to the same association and it is used by ROSE to convey the corresponding APDU on the proper association.

When a CL mode of operation is desired by the requesting OSMAP, the context establishment or release primitives are not required. In this case, the calling and called OSMAP addresses are contained in each M-operation primitive.

For synchronization of the operations GET, SET, COMPARE and ACTION a BLOCKING operation is provided. See 8.1.10.

The following general arguments are part of the common management service primitives described in 8.1.1 to 8.1.10.

- InvokeId:

  This argument specifies the identifier assigned to the operation unit (invocation). It can be used to distinguish the present operation unit from others that the invoking OSMAE may have in progress.

  It is passed to the ROS element to provide a connection between an invoke and its results or error when an association allows for overlapping operations. This argument may also be included in the user information parameter of ACSE association establishment and release service elements.

- ResourceId:

  This argument identifies the resource to which the management information pertains. It provides the context for the specific information in the PDUs. It may contain multiple identification attributes of the resource as described in the resource specification.

- AccessControl:

  This optional argument contains information of unspecified form to be used as input to access control functions in the initiating OSMAP and/or responding OSMAP.

- TestNumber:

  This optional parameter identifies the maintenance process to which the particular operation pertains. It preserves the overall context, even when the association at the application layer has been released. If several maintenance processes are concurrently active on the same association, it allows a given operation to be related to one of these processes. It is an optional parameter in the sense that the requesting OSMAP can use it or not. If it is present in a given INVOKE APDU, the responding OSMAP must return it unchanged in all the APDUs related to the initial request.

CMIS implicitly utilizes ROSE operation class 2 (asynchronous operation reporting success or failure) and association class 3 (only the association initiating application entity can invoke ordinary remote operations). The association class concept does not apply to the CL mode of operations.

**Primitive procedures overview**

The common processing to be executed at the responding OSMAE and OSMAP on reception of a management APDU is described below. Management operation specific processing is described later in dedicated sections.

When an executing OSMAE receives a management APDU, it performs the following checks:

- check the syntax of the received APDU,

- check the invokeId,

- check the operation code,

- in the case of an error, a REJECT APDU will be returned in each of the above checks.

On receiving the management operation indication from the OSMAE, the executing OSMAP performs the following common processing:

- check the ResourceId. An ERROR APDU is returned if the resource is unknown.

- if the AccessControl parameter is not required, no check is executed on it (not even if this parameter is present or absent in the received APDU).

- if the AccessControl parameter is required and it is absent or incorrect, an ERROR APDU will be returned.

No check is executed on the ExchangeId parameter.

### 8.1.1 Association Establishment (M-INITIALIZE)

Any OSMAP user (either a requesting OSMAP or an executing OSMAP) may initiate an association establishment by providing an M-INITIALIZE request service primitive to the OSMAE.

On receiving an M-INITIALIZE request service primitive from the OSMAP the OSMAE shall:

- form an M-INITIALIZE request APDU based on the parameters of the M-INITIALIZE request service primitive;

- send the APDU compiled as user data in the request form of a member of the set of context acquiring ACSE services. The context acquired shall be such that its properties are consistent with the M-INITIALIZE parameters. Details of these parameters are shown in Appendix C;

- enter the "context acquiring pending" state.

On receiving a confirm form of a member of the set of context acquiring ACSE service, with an M-INITIALIZE response APDU as user data while in the state "context acquiring pending", the OSMAE shall:

- if the result parameter of the ACSE primitive indicates a failure, issue an M-INITIALIZE confirm service primitive to the OSMAE user, with parameters derived from the ACSE confirm, including a diagnostic more severe than warning, and thereafter cease to exist;

- if the result parameter of the ACSE primitive indicates success, issue an M-INITIALIZE confirm service primitive to the OSMAE user, with parameters derived from the data items received and from the parameters received on the ACSE response primitive.

On receiving a confirm form of a member of the set of context acquiring ACSE service, without user data while in the state "context acquiring pending", and with a result parameter indicating failure, the OSMAE shall issue an M-INITIALIZE confirm service primitive to the OSMAE user with a diagnostic more severe than warning, and thereafter cease to exist.

### 8.1.2 Association Release

#### 8.1.2.1 Orderly Release (M-TERMINATE)

Either OSMAP user (either a requesting OSMAP or an executing OSMAP) may initiate an association release by providing an M-TERMINATE request service primitive to the OSMAE.

On receiving an M-TERMINATE request service primitive from the OSMAP user, the OSMAE shall:

- send an M-TERMINATE request PDU as user data on one of the set of context relinquishing ACSE services;

- enter the "context relinquishing pending" state.

On receiving an M-TERMINATE response PDU while in the state "context acquiring pending", the OSMAE shall:

- issue an M-TERMINATE confirm service primitive to the OSMAE user, with parameters derived from the data items received;

- thereafter cease to exist.

#### 8.1.2.2 Abrupt Release (M-ABORT)

Either OSMAP (the requesting and the executing) may initiate an abrupt association release by providing an M-ABORT request service primitive to the OSMAE.

On receiving an M-ABORT request service primitive from the OSMAE user, the OSMAE shall:

- construct an M-ABORT request PDU based on the parameters of the M-ABORT request service primitive;

- send the PDU as user data on the request form of a member of the set of context aborting ACSE services;

- thereafter cease to exist.

On receiving an indication form of the set of aborting ACSE services, a P-U-ABORT indication or a P-P-ABORT indication service primitive, the OSMAE shall:

- issue an M-ABORT indication service primitive to the OSMAE user, with parameters derived from the data items received;
- thereafter cease to exist.

On receiving an M-ABORT request PDU, the OSMAE shall:

- issue an M-ABORT indication service primitive to the OSMAE user, with parameters derived from the data items received;
- thereafter cease to exist.

On detecting a protocol error, an OSMAE shall:

- issue an M-ABORT indication service primitive to the OSMAE user, with diagnostic indicating protocol error;
- construct an M-ABORT request PDU with diagnostic indicating protocol error;
- send the PDU constructed on the request form of a member of the set of aborting ACSE services;
- thereafter cease to exist.

### 8.1.3 Invocation of a Specific Action (M-ACTION)

The function of the M-ACTION operation is to request the executing OSMAP to perform the action specified. This function provides a mechanism for initiating an action or causing state transitions within a resource.

#### 8.1.3.1 Arguments for Invoking a Management Action

- ActionType:

  This argument specifies a particular action that is to be performed. Besides the action types defined in this Technical Report (see above), there is the possibility to define other (private) action types to meet specific requirements.

- ActionArgument:

  This argument allows the specification of particular information related to a given action type (e.g. loop timer value). As for the action type, private specifications are possible.

- ResourceId:

  Defined in 8.1.

- TestNumber:

  Defined in 8.1.

- AccessControl:

  Defined in 8.1.

- ActionResult:

  This optional argument conveys information related to the outcome of the executed action.

- CurrentTime:

  This optional argument indicates the time at which the responding OSMAP completed the action.

- Status:

  The error status codes are:

  - noSuchOperation: the M-ACTION operation is not supported.
  - noSuchResource: the ResourceId specified is not recognized by the responding OSMAE.
  - accessDenied: the action was not performed due to the fact that the value of AccessControl was not acceptable or access was denied for other reasons.
  - noSuchAction: the ActionType specified is not supported.
  - noSuchActionArg: one or more of the specified action arguments are not supported.

#### 8.1.3.2 Procedures for Invoking a Management Action

A requesting OSMAP initiates an action operation in order to request an executing OSMAP to direct a resource to perform an action as specified for the particular resource type. The result of the action is returned to the requesting OSMAP as a result.

The executing OSMAP checks the ActionType to detect the requested action and ActionArgument if present.

#### 8.1.3.3 Specific Cases for Maintenance

These management actions are always executed after a context establishment phase. Therefore they are always preceded by the M-INITIALIZE primitive, specifying the required bearer (B or D) channel. The mapping of this service element into the network layer primitives depends on the presence (or absence) of a connection on the requested bearer channel and is detailed in 9.2.1.

The following actions are defined in this Technical Report:

- loop activation,
- loop deactivation,
- loop deactivation request,
- self-test request,
- self-test activation,
- self-test deactivation,
- self-test continuation,
- test call,
- start event report,
- stop event report.

### 8.1.4 Getting Attributes (M-GET-ATTRIBUTES)

The function of the M-GET-ATTRIBUTES operation is to request the executing OSMAP to read the specified resource attributes and to return them to the requesting OSMAP. The M-GET-ATTRIBUTES operation may specify more than one attribute of a single resource to be read.

#### 8.1.4.1 Arguments for Get Attributes

- AttributeList:

  This argument defines the parameter identifiers that have to be read by the executing OSMAP. Besides the attributes defined in this Technical Report, there is the possibility to define private attributes to meet particular requirements. The values of the attributes data elements are meaningless in the INVOKE APDU and are set to the read values in the corresponding RESULT APDU.

- Synchronization:

  This optional parameter indicates how the requesting OSMAP wants the executing OSMAP to synchronize the reading of multiple attributes.

- ResourceId:

  Defined in 8.1.

- TestNumber:

  Defined in 8.1.

- AccessControl:

  Defined in 8.1.

- CurrentTime:

  This optional argument indicates the time at which the responding OSMAP completed the reading of the attributes.

- Status:

  The error status codes are:

  - noSuchOperation: the M-GET operation is not supported.
  - noSuchResource: the ResourceId specified is not recognized by the responding OSMAP.
  - accessDenied: the attributes were not read due to the fact that the value of AccessControl was not acceptable or that the access was denied for other reasons.
  - syncNotSupported: the type of synchronization specified by the Synchronization parameter is not supported.
  - noSuchAttribute: the AttributeId specified is not supported.
  - getListError: one or more attributes were not read for one of the above error reasons; the values read are returned.

#### 8.1.4.2 Procedures For Getting Attributes

A requesting OSMAP sends an M-GET-ATTRIBUTES operation to an executing OSMAP to request the executing OSMAP to read the values of the specified data elements and to return those values to the requesting OSMAP.

The AttributeList is read according to the synchronization parameter. The executing OSMAP generates the appropriate response or error and returns it to the requesting OSMAP.

#### 8.1.4.3 Use of GET-ATTRIBUTES in the Context of Maintenance

Monitoring of remote resources and status enquiry are particular cases of the GET-ATTRIBUTES function.

### 8.1.5 Setting Remote Attributes (M-SET-ATTRIBUTES)

The purpose of the M-SET-ATTRIBUTES operation is to request the executing OSMAP to modify the specified resource attributes. The M-SET-ATTRIBUTES operation may specify more than one attribute.

#### 8.1.5.1 Arguments for Set Attributes (M-SET)

- AttributeList:

  This argument defines the parameter identifiers that have to be modified by the executing OSMAP. Besides the attributes defined in this Technical Report, there is the possibility to define private attributes to meet particular requirements.

  The values of the attributes data elements are set to the desired values in the INVOKE APDU.

- Synchronization:

  This optional parameter indicates how the requesting OSMAP wants the executing OSMAP to synchronize the modifications of multiple attributes.

- ResourceId:

  Defined in 8.1.

- TestNumber:

  Defined in 8.1.

- AccessControl:

  Defined in 8.1.

- CurrentTime:

  This optional argument indicates the time at which the responding OSMAP completed setting the attributes.

- Status:

  The error status codes are:

  - noSuchOperation: the M-SET operation is not supported.
  - noSuchResource: the ResourceId specified is not recognized by the responding OSMAP.
  - accessDenied: the attributes were not modified due to the fact that the value of AccessControl was not acceptable or that the access was denied for other reasons.
  - syncNotSupported: the type of synchronization specified by the Synchronization parameter is not supported.
  - noSuchAttribute: the AttributeId specified is not supported.
  - invalidAttributeValue: the attribute value specified is out of range or otherwise inappropriate.
  - setListError: One or more attributes were not set for one of the above error reasons; the attributes that could be set are indicated.

#### 8.1.5.2 Procedures for Setting Attributes

A requesting OSMAP sends an M-SET-ATTRIBUTES operation to an executing OSMAP to request it to modify the values of the specified data elements, replacing the current values with

the values specified in the attribute list.

The Attributes in the List are modified according to the synchronization parameter. The executing OSMAP generates the appropriate response or error and returns it to the requesting OSMAP.

#### 8.1.5.3 Use of SET-ATTRIBUTES in the Context of Maintenance

This function will be used to set, reset or update counters, thresholds and other parameters involved in the maintenance process.

### 8.1.6 Event Reporting (M-EVENT-REPORT, M-CONFIRMED-EVENT-REPORT)

The function of an event reporting operation is to notify to a remote OSMAP an event that has occurred in a reporting OSMAP. There are two types of event reporting operations: confirmed and non-confirmed. The invoke arguments in the two are identical. There is a result (acknowledgement) or error return for the confirmed, while nothing is returned for the non-confirmed operation.

#### 8.1.6.1 Arguments for Event Reporting

- ResourceId:

  Defined in 8.1.

- EventType:

  This parameter identifies the type of the event being reported. Besides the events defined in this Technical Report, there is the possibility to define private events to meet particular requirements.

- TestNumber:

  Defined in 8.1.

- EventTime:

  This field contains the time of generation of the event.

- EventInfo:

  This field identifies extra optional information that the reporting OSMAP is able to supply about the event. An event is defined as conditions that occur in relation to the use of resources in the open system of the reporting OSMAP. See the resource specification for definitions.

- EventAckInfo:

  This field identifies the success of the receipt of the Event PDU.

- Status:

  The error status codes are:

  - noSuchOperation: the M-CONFIRMED-EVENT-REPORT operation is not supported.

    *Note 4*

    *The reporting OSMAP must not deduce from such response that the OSMAP reported to has received the information.*

  - noSuchResource: the ResourceId specified is not recognized by the responding OSMAP.

- accessDenied: the responding OSMAP cannot accept the event due to access control restrictions.
- eventFiltered: the event was not logged due to a filtering operation at the responding OSMAP (the details of event filtering are for further study).
- noSuchAttribute: the EventInfo specified is not supported.
- invalidAttributeValue: the attribute value specified is out of range or otherwise inappropriate.
- noLog: event has not been logged due to lack of system facilities.

#### 8.1.6.2 Procedures for Event Reporting

An M-CONFIRMED-EVENT-REPORT operation or M-EVENT-REPORT operation is initiated by a reporting OSMAP whenever a particular change of state occurs that has been selected as an event to be reported. They are sent to a remote OSMAP and they are unsolicited messages in the sense that the OSMAP to which the event is reported did not make specific requests for them.

When the event occurs, the reporting OSMAP collects the information associated with the event.

If the event report arrives in an M-CONFIRMED-EVENT-REPORT operation, an M-CONFIRMED-EVENT-REPORT result or error primitive will be used by the OSMAP, to which the event is reported, to acknowledge it to the reporting OSMAP.

### 8.1.7 Comparing Attribute Values (M-COMPARE)

This function is used to compare the value of an attribute to that of a user provided value. The result is a status of success or failure which may be used by the blocking operation to determine whether or not to proceed with other operations (e.g. set, action, ...).

The COMPARE operation generates a RESULT APDU if the comparison was executed and has given the expected result as indicated in the CompareOperat parameter. An ERROR APDU is generated either if the comparison could not be executed because of an error or if the comparison was executed, but did not give the expected result (e.g. the compare operator specified was "equal", the comparison value was X, but the current attribute value to be compared was not equal to X).

#### 8.1.7.1 Arguments

- ResourceId:

  Defined in 8.1.

- CompareAtt:

  This parameter defines the attribute to be compared and the value to be used in the comparison.

- CompareOperat:

  This parameter indicates the operator to be used in the comparison. The first operand is always the attribute value read in the executing OSMAP database; the second operand is the value contained in the CompareAtt parameter of the M-COMPARE APDU.

- Status:

  The error status codes are:

noSuchOperation
noSuchResource
accessDenied
noSuchAttribute
invalidAttributeValue
noSuchOperator
compareFailure: the two operands do not satisfy the COMPARE operation.

- TestNumber

  Defined in 8.1.

- AccessControl

  Defined in 8.1.

#### 8.1.7.2 Procedures for Comparing Attributes

This operation is used by the requesting OSMAP to determine the relative value of an attribute to a provided value.

The Compare operation generates a positive result APDU if the attribute value in the executing OSMAP and the value indicated in the compare APDU satisfy the comparison operation. When the two operands do not satisfy the compare operation an ERROR APDU is generated with the error code "compare failure".

The operation is intended to be primarily used within a blocking operation to provide a condition for preventing the GET, SET or ACTION operations to be performed if not appropriate. In this context the "Stop on error" synchronization will be used to prevent the requesting OSMAP from executing the operations following the "compare" if the comparison did not give the expected result.

### 8.1.8 Blocking Operation (M-BLOCKING)

This function provides a means for synchronizing operations such as GET, SET, COMPARE and ACTION at the executing side. The use of an operation will cause the requesting side to group the operations into a single (global) request. In doing so, it allows the responder to execute multiple operations as a group. It allows synchronization of operations on different resources as well as of different type of operations on the same resource.

On completion of the global request, each of the individual results will be returned in a global result.

#### 8.1.8.1 Argument for Blocking Primitives

- OperationList:

  This parameter contains one or more of the possible operations (GET, SET, ACTION and/or COMPARE) with all their appropriate arguments. The operations may occur in any combination, including repeated instances of one type.

- TestNumber:

  Defined in 8.1.

- AccessControl:

  Defined in 8.1.

- Synchronization:

  This optional parameter indicates how the requesting OSMAP wants the executing OSMAP to synchronize the execution of the various operations.

- OperationResults:

  This field contains any resulting information associated with the individual operations.

- Status:

  The error status codes are:

  - accessDenied: (same as for other functions).
  - noSuchOperation: the operation specified is not supported.
  - SyncNotSupported: the type of sync specified by the sync parameter is not supported.
  - BlockListError: one or more operations returned on error; successful operations are returned.

#### 8.1.8.2 Procedures for Blocking Primitives

The blocking operation is used by the requestor to request an executor to perform a set of SET, GET, COMPARE and/or ACTION operations. There may be more than one instance of each type of operation and the types may appear in any combination. This allows the synchronization of operations at the executing side and further requests will not be processed until completion of the blocked operations.

If the blocking APDU defines an unsupported operation, a blocking error will be returned. If the operation is valid, the request will be interpreted according to the semantics defined for that operation. The check of operation types is continued until all of the requests have been interpreted. The execution of the operations proceeds in accordance with the information contained in the synchronization argument.

The results of all successful operations from the blocking list are concatenated and returned in the blocking result. The individual results are concatenated in the same order as requested for by the execution.

### 8.1.9 Comprehensive Overview of M-Primitive Arguments

Tables 3 and 4 provide a comprehensive overview of the M-primitive parameters described above. Table 3 indicates the parameters, while Table 4 lists the error codes.

| | INITIATE | ACTION | GET | SET | EVENT REP. | TERMINATE | ABORT | BLOCKING | COMPARE |
|---|---|---|---|---|---|---|---|---|---|
| TestNumber | | O | O | O | O | | | O | O |
| Access Contr | O | O | O | O | | O | O | O | O |
| CallingOSMAP | * | *Note 5* | *Note 5* | *Note 5* | *Note 5* | | | *Note 5* | *Note 5* |
| CalledOSMAP | * | *Note 5* | *Note 5* | *Note 5* | *Note 5* | | | *Note 5* | *Note 5* |
| ResourceId | | * | * | * | * | | | | * |
| AssociationId | | *Note 5* | *Note 5* | *Note 5* | *Note 5* | * | * | *Note 5* | *Note 5* |
| ActionType | | * | | | | | | | |
| ActionResult | | O | | | | | | | |
| AttributeList | | | * | * | | | | | |
| EventType | | | | | * | | | | |
| CurrentEvent/ CurrentTime | | O | O | O | * | | | | |
| EventInfo | | | | | O | | | | |
| EventAckInfo | | | | | O | | | | |
| CompareOp | | | | | | | | | * |
| SyncInfo | | | O | O | | | | O | |
| ActionArgument | | O | | | | | | | |
| CompareAtt | | | | | | | | | * |
| OperationList | | | | | | | | * | |
| OperationResult | | | | | | | | * | |

Legend: Keys for the use of parameters are: * = mandatory, O = optional.

**Table 3 - M-Primitive Arguments**

*Note 5*

*The Association Reference parameter is present only in the connection-oriented mode of operation. It is not present in the connectionless mode of operation. In this latter case it is replaced by the calling and called OSMAP addresses.*

| | INITIATE | ACTION | GET | SET | EVENT | TERMINATE | ABORT | BLOCKING | COMPARE |
|---|---|---|---|---|---|---|---|---|---|
| AccessDenied | * | * | * | * | | * | * | * | * |
| NoSuchResource | | * | * | * | * | | | | * |
| NoSuchOperation | | * | * | * | * | * | | * | * |
| EventFiltered | | | | | * | | | | |
| NoSuchAttrib | | | * | * | * | | | | * |
| NoSuchActArgt/Attrib | | * | | | | | | | |
| InvalidAttributeValue | | | | * | * | | | | * |
| SyncNotSupported | | | * | * | | | | * | |
| SetListError | | | | * | | | | | |
| GetListError | | | * | | | | | | |
| NoSuchOperator | | | | | | | | | * |
| CompareFailure | | | | | | | | | * |
| NoSuchAction | | * | | | | | | | |
| NoLog | | | | | * | | | | |

**Table 4 - Error Codes**

## 8.2 Use of Remote Operation Services

### 8.2.1 General

The CMIS uses ROSE as defined by X.ROS. These services are obtained by the ROSE provider entering an exchange of Application Protocol Data Units (APDUs). The services and invocations can be described by means of the CCITT ROSE macros.

An overview of ROSE is given below, while 8.3 contains the abstract description.

Four APDUs are defined:

Invoke APDU,
Return Result APDU,
Return Error APDU,
Reject APDU.

The binary representation of each APDU can be inferred, by using the ASN.1 notation as per ISO DIS 8824 and ISO DIS 8825 (see Appendix B).

ROSE is mainly composed of:

- RO-INVOKE: to request an operation to be performed by the remote OSMAE. This service is mapped into the INVOKE APDU.
- RO-RESULT: to allow an OSMAE to return a positive reply on a successfully performed operation. It is mapped into the RESULT APDU.
- RO-ERROR: to allow an OSMAE to return a negative reply on an unsuccessfully performed operation. It is mapped into the ERROR APDU.
- RO-REJECT-U: to enable an OSMAE to reject a request that is unacceptable. It is mapped into the REJECT APDU.
- RO-REJECT-P: to allow the RO service provider to notify the RO service user (CMIS) that it detected a problem. It is also mapped into REJECT APDU.

### 8.2.2 Mapping of M-Primitives into ROSE

Each of the M-type primitives (REQUEST, INDICATION, RESPONSE, CONFIRM) shall be mapped into one of the ROS elements and the primitive parameters will ultimately be coded into one of the APDU elements.

Two types of mapping exist. Their use depends on whether a confirmed or a non-confirmed operation is initiated by the requestor. Table 5 shows the confirmed operation which applies, for example, to actions like control of loops, self-tests and test calls.

| M-Primitives (SMI) (Note 6, 7) | ROSE Primitives | | | | | | | | DATA (SMDSI) (Note 8) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | INVOKE | | These Service Elements occur alternatively only | | | | | | | |
| | | | RESULT | | ERROR | | REJECT | | | |
| | REQ | IND | REQ | IND | REQ | IND | REQ | IND | REQ | IND |
| REQ (R) | x | | | | | | | | x | |
| IND (E) | | x | | | | | | | | (Note 9) |
| RSP (E) | | | x | | x | | x | | x | |
| CNF (R) | | | | x | | x | | x | | x |

**Table 5 - Mapping of M-Primitives into ROSE for confirmed Transactions**

*Note 6*

*For M-GET, M-SET, M-COMPARE and M-CONFIRMED-EVENT-REPORT the same pattern applies.*

*Note 7*

*(R) indicates a primitive issued by the requestor, (E) indicates a primitive issued by the executor.*

*Note 8*

*The indication "DATA" is used for both DATA and UNIT-DATA lower layer service elements.*

*Note 9*

*This DATA IND may not be received if the PDU is rejected by ROSE (RO-REJ-P).*

Table 6 shows the non-confirmed operation, which applies, for example, to reports on loop deactivation, self-tests, alarms and the desire to release a connection.

The APDUs will ultimately be carried by means of the lower layer DATA service elements (REQUEST, INDICATION).

| M-Primitives (SMI) (Note 10) | ROSE Primitives | | | | | | | | DATA (SMDSI) (Note 11) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | INVOKE | | These Service Elements occur alternatively only | | | | | | | |
| | | | RESULT | | ERROR | | REJECT | | | |
| | REQ | IND | REQ | IND | REQ | IND | REQ | IND | REQ | IND |
| REQ (R) | x | | | | | | | | x | |
| IND (E) | | x | | | | | | x | | x |

**Table 6 - Mapping of M-Primitives into ROSE for non-confirmed Transactions**

*Note 10*

*(R) indicates a primitive issued by the requestor, (E) indicates a primitive issued by the executor.*

*Note 11*

*The indication "DATA" is used for both DATA and UNIT-DATA lower layer service elements.*

## 8.3 APDU Formats and Codes

The APDUs as interchanged between peer OSMAEs are defined by means of the ASN.1 notation (see ISO DIS 8824). The ASN.1 syntax is briefly presented in Appendix D.

Examples of the resulting formats are given in Appendix E.

A tutorial representation of the APDUs is given in 8.3.1, while the detailed definition of the protocol is given in 8.3.2.

### 8.3.1 Representation of the APDUs

Figures 20 to 24 provide a tutorial representation of the structure of various Invoke/Result APDUs chosen as examples.

The formal definition of each APDU is given in Appendix B.

```
APDU Identifier (= 1 "INVOKE")
    SEQUENCE Identifier
         Invoke ID
         Linked ID (Optional)
         Operation (= 5 "ACTION")
         SEQUENCE Identifier (ACTION INVOKE Parameters)
                SEQUENCE Identifier (ResourceID)
                         Resource Class
                         Resource Instance              (Optional)
                Access Control                          (Optional)
                Action Type
                Action Argument                         (Optional)
                Test ID                                 (Optional)
```

**Figure 20 - INVOKE ACTION Operation**

```
APDU Identifier (= 2 "RESULT")
    SEQUENCE Identifier
         Invoke ID
         SEQUENCE Identifier (ACTION RESULT Parameters)
                Current Time                            (Optional)
                Action Result                           (Optional)
```

**Figure 21 - RESULT ACTION Operation**

```
APDU Identifier (= 1 "INVOKE")
    SEQUENCE Identifier
         Invoke ID
         Operation (= 3 "GET")
         SEQUENCE Identifier (GET INVOKE Parameters)
                SEQUENCE Identifier (ResourceID)
                         Resource Class
                         Resource Instance              (Optional)
                Access Control                          (Optional)
                Synchronization                         (Optional)
                Attribute ID List
                Test ID                                 (Optional)
```

**Figure 22 - INVOKE GET Operation**

```
APDU Identifier (= 1 "INVOKE")
    SEQUENCE Identifier
         Invoke ID
         Operation (= 6 "COMPARE")
         SEQUENCE Identifier (COMPARE Parameters)
                SEQUENCE Identifier (ResourceID)
                         Resource Class
                         Resource Instance              (Optional)
                Access Control                          (Optional)
                Compare Attribute
                Compare Operator
                Test ID                                 (Optional)
```

**Figure 23 - INVOKE COMPARE Operation**

```
APDU Identifier (= 1 "INVOKE")
    SEQUENCE Identifier
         Invoke ID
         Linked ID                                      (Optional)
         Operation (= 1 "EVENT REPORT")
         SEQUENCE Identifier (EVENT REPORT Parameters)
                SEQUENCE Identifier (ResourceID)
                         Resource Class
                         Resource Instance              (Optional)
                EventType
                Event Time
                Event Info                              (Optional)
                Test ID                                 (Optional)
```

**Figure 24 - INVOKE EVENT REPORT Operation**

### 8.3.2 ISDN Common Management Information Protocol (CMIP)

The ECMA definitions for the ISDN Management Information Protocol elements are presented in Figures 25 to 79.

```
ECMA-CMIP-APDUs  DEFINITIONS ::= BEGIN

    IMPORTS BIND, UNBIND, OPERATION, ERROR from
       Remote-Operation-Notation
       {joint-iso-ccitt remoteOperations (4), notation (0) }

    IMPORTS APPLICATION-SERVICE-ELEMENT, APPLICATION-CONTEXT FROM
       Remote-Operation-Notation-Extension
       {joint-iso-ccitt remoteOperations (4), notation-extension (2) }

    IMPORTS rOSE,InvokeIDType from
       Remote-Operation-APDUs
       {joint-iso-ccitt remoteOperations (4), apdus (1) }

    IMPORTS aCSE from X.ACP
```

**Figure 25 - CMIP APDUs**

```
ECMA-CMIP  APPLICATION-CONTEXT
            APPLICATION SERVICE ELEMENTS { aCSE }
            BIND initialize
            UNBIND terminate
            REMOTE OPERATIONS { rOSE}
            OPERATIONS OF { iSDmntAse}
            ABSTRACT SYNTAXES { OBJECT IDENTIFIER for ASN.1 }
            :: = {iso (1),identified-organization (3),
                  ICD (Tbd),
                  organization code (Tbd) }
```

**Figure 26 - ISDN Maintenance Application Context Definition**

```
iSDNmntAse APPLICATION-SERVICE-ELEMENT
               OPERATIONS { initialize, terminate, confirmedEventReport,
               eventReport, get, set, action, compare, blocking }
       ::= { OBJECT IDENTIFIER for iSDNmntAse }
```

**Figure 27 - ISDN Maintenance Application Service Element Definition**

```
initialize BIND
   ARGUMENT SEQUENCE {
            accessControl AccessControl OPTIONAL,
            maintenanceProcessNumber INTEGER OPTIONAL }
   RESULT NULL
   BIND-ERROR { accessDenied }
   ::= NULL
```

**Figure 28 - Bind Operation (M-INITIALIZE)**

```
terminate UNBIND
   ::= NULL
```

**Figure 29 - Unbind Operation (M-TERMINATE, M-ABORT)**

```
confirmedEventReport OPERATION
   ARGUMENT SEQUENCE {
            resourceID      ResourceID,
            eventType       EventType,
            eventTime       TimeStamp,
            eventInfo       EventInfo OPTIONAL,
            testID          TestID OPTIONAL }
   RESULT   eventAckInfo  NULL
   ERRORS   {noSuchOperation, noSuchResource, accessDenied,
            eventFiltered, noSuchAttribute, invalidAttributeValue,
            nolog }
   ::= 1
```

**Figure 30 - Confirmed Event Report Operation (M-CONFIRMED-EVENT-REPORT)**

```
eventReport OPERATION
   ARGUMENT SEQUENCE {
            resourceID      ResourceID,
            eventType       EventType,
            eventTime       TimeStamp,
            eventInfo       EventInfo OPTIONAL,
            testID          TestID OPTIONAL }
   ::= 2
```

**Figure 31 - Unconfirmed Event Report Operation (M-EVENT-REPORT)**

```
get OPERATION
   ARGUMENT SEQUENCE {
            resourceID      ResourceID,
            accessControl   AccessControl OPTIONAL,
            synchronization CMISSync OPTIONAL DEFAULT (0),
            attributeIdList SET OF Parameters,
            testID          TestID OPTIONAL }
   RESULT   SEQUENCE {
            currentTime     TimeStamp OPTIONAL,
            attributeList   SET OF MISParameter }

   ERRORS   {noSuchOperation, noSuchResource, accessDenied,
            syncNotSupported, noSuchAttribute, getListError}
   ::= 3
```

**Figure 32 - Get Operation (M-GET)**

```
set OPERATION
   ARGUMENT SEQUENCE {
            resourceID       ResourceId
            accessControl    AccessControl OPTIONAL,
            synchronization  CMISSync OPTIONAL DEFAULT (0),
            attributeList    SET OF MISParameter,
            testID           TestID OPTIONAL }
   RESULT   SEQUENCE {
            currentTime      TimeStamp OPTIONAL,
            attributeList    SET OF MISParameter }
   ERRORS   {noSuchOperation, noSuchResource, accessDenied,
            syncNotSupported, noSuchAttribute, setListError,
            invalidAttributeValue}
   ::= 4
```

**Figure 33 - Set Operation (M-SET)**

```
action OPERATION
   ARGUMENT SEQUENCE {
            resourceID       ResourceID,
            accessControl    AccessControl OPTIONAL,
            actionType       ActionType,
            actionArg        SET OF MISParameter,
            testID           TestID OPTIONAL }
   RESULT   SEQUENCE {
            currentTime      TimeStamp OPTIONAL,
            actionResult     MISParameter }
   ERRORS   {noSuchOperation, noSuchResource, accessDenied,
            noSuchAction,noSuchActionArg }
   LINKED { action, eventReport, confirmedEventReport }
   ::= 5
```

**Figure 34 - Action Operation (M-ACTION)**

```
compare OPERATION
   ARGUMENT SEQUENCE {
            resourceID       ResourceID,
            accessControl    AccessControl OPTIONAL,
            compareAtt       SET OF MISParameter,
            compareOperat    CompareOperat,
            testID           TestID OPTIONAL }
   RESULT   NULL
   ERRORS   {noSuchOperation, accessDenied, noSuchResource,
            noSuchAttribute, invalidAttributeValue, noSuchOperator,
            compareFailure }
   ::= 6
```

**Figure 35 - Compare Operation (M-COMPARE)**

```
blocking OPERATION
   ARGUMENT SEQUENCE {
            accessControl   AccessControl OPTIONAL,
            synchronization CMISSync OPTIONAL DEFAULT (0),
            operationList SEQUENCE OF Invoke }
   RESULT  SET OF CHOICE { ReturnResult, ReturnError }
   ERRORS  {noSuchOperation, accessDenied, syncNotSupported,
            blockListError }
   ::= 7
```

**Figure 36 - Blocking Operation (M-BLOCKING)**

```
noSuchOperation ERROR
   PARAMETER operationCode INTEGER          -- range 1-127
   ::= 1

noSuchResource ERROR
   PARAMETER resourceID ResourceID
   ::= 2

accessDenied ERROR
   PARAMETER accessGranted BITSTRING {
           none (0), get (1), set (2), action (3), log (4) }
   ::= 3


eventFiltered ERROR
   PARAMETER eventFiltered EventType
   ::= 4

noSuchAttribute ERROR
   PARAMETER attribute CHOICE { EventType, Parameters }
   ::= 5

invalidAttributeValue ERROR
   PARAMETER attributeValue MISParameter
   ::= 6

syncNotSupported ERROR
   PARAMETER syncAvailable CMISSync
   ::= 7

getListError ERROR
   PARAMETER getAttributeStatusList SET OF MISGetParameterStatus
   ::= 8

MISGetParameterStatus ::= CHOICE {
        misParameterIdError     [0] IMPLICIT MISParameterIdError,
        misParameter            [1] IMPLICIT MISParameter }

MISParameterIdError ::= SEQUENCE {
        errorStatus     [0] IMPLICIT ErrorStatus,
        misParameterId  [1] IMPLICIT Parameters }
```

**Figure 37 - CMIP Error Definitions**

*(continuation)*

```
setListError ERROR
   PARAMETER setAttributeStatusList SET OF MISSetParameterStatus
   ::= 9

   MISSetParameterStatus ::= CHOICE {
           misParameterError       [0] IMPLICIT MISParameterError,
           misParameter            [1] IMPLICIT MISParameter }

   MISParameterError ::= SEQUENCE {
           errorStatus     [0] IMPLICIT ErrorStatus,
           misParameter    [1] IMPLICIT MISParameter }

      ErrorStatus ::= INTEGER {
              accessDenied    (1),
              noSuchOperation (2),
              noSuchAttribute (3),
              invalidAttributeValue (4) }

   noSuchAction ERROR
      PARAMETER actionType ActionType
      ::= 10

   noSuchActionArgument ERROR
      PARAMETER actionArg ActionArgValue
      ::= 11

   noSuchOperator ERROR
      PARAMETER compareOperat INTEGER
      ::= 12

   compareFailure ERROR
      PARAMETER compareAtt MISParameter
      ::= 13
   blockListError ERROR
      PARAMETER blockOperationStatusList
              SEQUENCE OF BlockOperationStatus
      ::= 14

   nolog ERROR
      PARAMETER NULL
      ::= 15

      BlockOperationStatus ::= CHOICE {
                      [2] IMPLICIT ReturnResult,
                      [3] IMPLICIT ReturnError,
                      [4] IMPLICIT Reject }
```

**Figure 38 - CMIP Error Definitions**

```
ResourceID ::= SEQUENCE {
          ResourceClass, ResourceInstance OPTIONAL }

ResourceClass ::= OBJECT IDENTIFIER

          -- The OBJECT IDENTIFIER Value for ECMA ISDN is

             { iso (1),
               identified organization (3),
               ICD for ECMA (Tbd),
               organization code (Tbd = ISDN Access),
               channel identifier
                       { D-Channel (255)
                         Associated B-Channel (0)      -- when the executor's
                                                          B-channel is unknown to
                                                          the requestor

                         B1 (1)
                         B2 (2)
                         B3 (3)
                            .
                            .
                            .
                         B31 (31) }
               service identifier
                  either D Channel
                        { Signalling      (0)
                          Frame           (1)
                          Packet          (2) }
                  or B Channels
                        { Circuit         (0)
                          Frame           (1)
                          Packet          (2) }
               layer identifier
                       { Layer 1          (1)
                         Layer 2          (2)
                         Layer 3          (3) } }

ResourceInstance ::= SEQUENCE {
          accessId        [0] IMPLICIT INTEGER OPTIONAL,

          -- the accessId is used to identify one out of several
          -- ISDN accesses available at a given endpoint.
               -- It could be equal to the network addr
               CESP            [1] IMPLICIT OCTETSTRING OPTION
     EventType  ::= CHOICE {
               privateEvent [0] ANY,    -- defined in private protocol
     nedEvent [1] IMPLICIT DefinedEvent }
```

**Figure 39 - CMIP Parameter Definitions**

```
DefinedEvent ::= INTEGER {
          StatusChange     (1),
          LossOfFraming    (2),
          LossOfSynch      (3) }
```

**Figure 40 - Events defined for the ISDN Access**

```
DefinedEvent ::= INTEGER {
          StatusChange     (1) }
```

**Figure 41 - Events defined for the D Channel globally (i.e. layer independently)**

```
DefinedEvent ::= INTEGER {
          FCSFailure       (1)
          LossOfFraming    (2) }
```

**Figure 42 - Events defined for the D Channel Layer 1**

```
DefinedEvent ::= INTEGER {
          UnsolicResp      (1),
          PeerReset        (2),
          StatusEnquiry    (3),
          N(R)Error        (4),
          FRMRRcvd         (5),
          UnknownFrame     (6),
          InvalidIFrame    (7),
          Wrongsize        (8),
          N201Error        (9),
          SupervisionTimeout (10) }
```

**Figure 43 - Events defined for the D Channel Layer 2**

```
DefinedEvent ::= INTEGER {
          N303Error              (1),
          N30YError              (2),
          OutCallConnected       (3),
          OutCallFailed          (4),
          OutCallDisconnected    (5),
          InCallConnected        (6),
          InCallFailed           (7),
          InCallDisconnected     (8),
          LoopTimeout            (9) }
```

**Figure 44 - Events defined for the D Channel Layer 3**

```
DefinedEvent ::= INTEGER {
          StatusChange     (1) }
```

**Figure 45 - Events defined for each B Channel globally (i.e. layer independently)**

```
DefinedEvent ::= INTEGER {
          LoopTimeout      (1),
          LossOfSynch      (2) }
```

**Figure 46 - Events defined for each B Channel Layer 1 - Circuit mode**

To be defined in the future Standard

**Figure 47 - Events defined for each B Channel Layer 1 - Frame mode**

To be defined in the future Standard

**Figure 48 - Events defined for each B Channel Layer 2 - Frame mode**

To be defined in the future Standard

**Figure 49 - Events defined for each B Channel Layer 1 - Packet mode**

To be defined in the future Standard

**Figure 50 - Events defined for each B Channel Layer 2 - Packet mode**

To be defined in the future Standard

**Figure 51 - Events defined for each B Channel Layer 3 - Packet mode**

The INTEGER Test Id is composed of two nibbles, the first nibble indicating the maintenance Process Number (Test Number) and the second indicating the Transaction Number (within a given process number). If it is present in an invocation request, it must be returned unchanged by the executor in all the APDUs related to the invoked operation.

**Figure 52 - TestIdentity**

```
AccessControl ::= CHOICE {
   privateAccessControl     [Application 0] EXTERNAL,
                            -- defined in private protocol
   definedAccessControl     [Application 1] IMPLICIT OCTETSTRING }
                            -- 16 octets max
```

**Figure 53 - AccessControl**

```
TimeStamp ::= CHOICE {
          generalisedTime GeneralisedTime,
          utcTime UtcTime,
          localTime EXTERNAL }
```

**Figure 54 - TimeStamp**

```
EventInfo ::= CHOICE {
          privateEventInfo   [2] ANY,
                          -- defined in private protocol
          definedEventInfo   [3] DefinedEventInfo }
```

**Figure 55 - EventInfo**

```
DefinedEventInfo ::= CHOICE {
          ctrlField          [0] IMPLICIT BITSTRING,
          status             [1] IMPLICIT ResourceStatusValue,
          CallInfo           [2] ANY }    -- layer specific
```

**Figure 56 - DefinedEventInfo**

```
CMISSync ::= [Application 2] IMPLICIT INTEGER {
          bestEffort      (0),
          ordered         (1),
          stopOnError     (2),
          atomic          (3) }
```

**Figure 57 - CMISSync**

*Note 12*

*The meanings of the synchronization parameters are:*

*bestEffort: All operations that can be executed will be executed, and those which cannot be executed will not be executed. The ordering is not important.*

*ordered: All operations that can be executed will be executed, and those which cannot be executed will not be executed. Operations are attempted in the sequence in which they are presented in the APDU.*

*stopOnError: The operations are processed sequentially in the order in which they appear in the APDU. However, when an unsuccessfull operation is encountered the remaining operations will not be attempted.*

*atomic: All operations are checked on whether they can be executed successfully. Either all or none will be executed.*

```
MISParameter ::= CHOICE
          { [7] IMPLICIT SET { Parameters },
                SEQUENCE { MISParameterId, MISParameterValue } }
```

**Figure 58 - MISParameter**

```
MISParameterId ::= CHOICE {
          privateParameterId [0] ANY,
          definedParameterId [1] IMPLICIT DefinedParameterId }
```

**Figure 59 - MISParameterId**

```
misParameterValue ::= CHOICE {
          privateValue          [0] ANY,
          protocolValue         [1] IMPLICIT INTEGER,
          resourceStatusValue   [2] IMPLICIT ResourceStatusValue,
          actionArgValue        [3] ActionArgValue }
```

**Figure 60 - MISParameterValue**

The MISParameter is defined as a choice between an "IMPLICIT SET" and a"SEQUENCE". The major aspects of the IMPLICIT SET can be summarized as follows:

- Contents of each parameter set to ZERO for GET Operation;
- Contents of each parameter set to its value for SET Operation;
- Contents of each parameter set to the comparison value for COMPARE Operation;
- Timer values are expressed and returned in tenths of seconds;
- A bit set to ONE in the EventReportMask indicates that the event value corresponding to that bit position must be reported;
- A bit set to ZERO in the EventReportMask indicates that the event value corresponding to that bit position must not be reported;
- Bit positions are numbered starting from ZERO;
- For example, to get event report only on loss of Synch (code: 3) EventReportMask should be equal to 8 (bit 3 = ONE).

The ASN.1 description for this case is shown in Figures 61 to 72, while Figure 73 shows the general structure for the SEQUENCE case.

```
        privateAttribute        [0] ANY OPTIONAL,
                        -- defined in private protocol
        resourceStatusValue     [1] IMPLICIT ResourceStatusValue
                                        OPTIONAL,
        loopReference           [2] IMPLICIT LoopReference
                                        OPTIONAL,
        loopTimer               [3] IMPLICIT INTEGER OPTIONAL,
                                        -- Timer unit: 5 seconds
        eventReportMask         [4] IMPLICIT BITSTRING OPTIONAL,
        lossOfFramingCounter    [5] IMPLICIT INTEGER OPTIONAL,
        lossOfSynchCounter      [6] IMPLICIT INTEGER OPTIONAL }
```

**Figure 61 - Parameters defined for the ISDN Access**

```
Parameters ::= {
        privateAttribute        [0] ANY OPTIONAL,
                        -- defined in private protocol
        eventReportMask         [1] IMPLICIT BITSTRING OPTIONAL,
        resourceStatusValue     [2] IMPLICIT ResourceStatusValue
                                        OPTIONAL }
```

**Figure 62 - Parameters defined globally for the D Channel (i.e. layer independently)**

```
Parameters ::= {
        privateAttribute        [0] ANY OPTIONAL,
                                -- defined in private protocol
        loopReference           [1] IMPLICIT LoopReference
                                        OPTIONAL,
        loopTimer               [2] IMPLICIT INTEGER OPTIONAL,
                                        -- Timer unit: 5 seconds
        eventReportMask         [3] IMPLICIT BITSTRING OPTIONAL,
        fcsFailureCounter       [4] IMPLICIT INTEGER OPTIONAL,
        lossOfFramingCounter    [5] IMPLICIT INTEGER OPTIONAL }
```

**Figure 63 - Parameters defined for the D Channel Layer 1**

```
Parameters ::= {
        privateAttribute        [0] ANY OPTIONAL,
                                -- defined in private protocol
        t200                    [1] IMPLICIT INTEGER OPTIONAL,
        n200                    [2] IMPLICIT INTEGER OPTIONAL,
        n201                    [3] IMPLICIT INTEGER OPTIONAL,
        n202                    [4] IMPLICIT INTEGER OPTIONAL,
        k                       [5] IMPLICIT INTEGER OPTIONAL,
        t201                    [6] IMPLICIT INTEGER OPTIONAL,
        t202                    [7] IMPLICIT INTEGER OPTIONAL,
        t203                    [8] IMPLICIT INTEGER OPTIONAL,
        loopReference           [9] IMPLICIT LoopReference OPTIONAL,
        loopTimer               [10] IMPLICIT INTEGER OPTIONAL,
                                        -- Timer unit: 5 seconds
        eventReportMask         [11] IMPLICIT BITSTRING OPTIONAL,
        unsolicRespCounter      [12] IMPLICIT INTEGER OPTIONAL,
        peerResetCounter        [13] IMPLICIT INTEGER OPTIONAL,
        statusEnquiryCounter    [14] IMPLICIT INTEGER OPTIONAL,
        n(R)ErrorCounter        [15] IMPLICIT INTEGER OPTIONAL,
        fRMRcvdCounter          [16] IMPLICIT INTEGER OPTIONAL,
        unknownFrameCounter     [17] IMPLICIT INTEGER OPTIONAL,
        invalidIFrameCounter    [18] IMPLICIT INTEGER OPTIONAL,
        wrongsizeCounter        [19] IMPLICIT INTEGER OPTIONAL,
        n201ErrorCounter        [20] IMPLICIT INTEGER OPTIONAL,
        supervisionTimeoutCounter [21] IMPLICIT INTEGER OPTIONAL }
```

**Figure 64 - Parameters defined for the D Channel Layer 2**

```
Parameters ::= {
          privateAttribute              [0] ANY OPTIONAL,
                                        -- defined in private protocol
           t303                         [1] IMPLICIT INTEGER OPTIONAL,
           t305                         [2] IMPLICIT INTEGER OPTIONAL,
           t308                         [3] IMPLICIT INTEGER OPTIONAL,
           t309                         [4] IMPLICIT INTEGER OPTIONAL,
           t310                         [5] IMPLICIT INTEGER OPTIONAL,
           t30Y                         [6] IMPLICIT INTEGER OPTIONAL,
           n303                         [7] IMPLICIT INTEGER OPTIONAL,
           n30Y                         [8] IMPLICIT INTEGER OPTIONAL,
           loopReference                [9] IMPLICIT LoopReference OPTIONAL,
           loopTimer                    [10] IMPLICIT INTEGER OPTIONAL,
                                        -- Timer unit: 5 seconds
           eventReportMask              [11] IMPLICIT BITSTRING OPTIONAL,
           n303ErrorCounter             [12] IMPLICIT INTEGER OPTIONAL,
           n30YErrorCounter             [13] IMPLICIT INTEGER OPTIONAL,
           outCallConnectedCounter      [14] IMPLICIT INTEGER OPTIONAL,
           outCallFailedCounter         [15] IMPLICIT INTEGER OPTIONAL,
           outCallDisconnectedCounter   [16] IMPLICIT INTEGER OPTIONAL,
           inCallConnectedCounter       [17] IMPLICIT INTEGER OPTIONAL,
           inCallFailedCounter          [18] IMPLICIT INTEGER OPTIONAL,
           inCallDisconnectedCounter    [19] IMPLICIT INTEGER OPTIONAL,
           loopTimeoutCounter           [20] IMPLICIT INTEGER OPTIONAL }
```

**Figure 65 - Parameters defined for the D Channel Layer 3**

```
Parameters ::= {
          privateAttribute        [0] ANY OPTIONAL,
                                  -- defined in private protocol
          eventReportMask         [1] IMPLICIT BITSTRING
                                      OPTIONAL,
          resourceStatusValue     [2] IMPLICIT ResourceStatusValue
                                      OPTIONAL }
```

**Figure 66 - Parameters defined for each B Channel globally (i.e. layer independently)**

```
Parameters ::= {
          privateAttribute        [0] ANY OPTIONAL,
                                  -- defined in private protocol
          loopReference           [1] IMPLICIT LoopReference
                                      OPTIONAL,
          loopTimer               [2] IMPLICIT INTEGER OPTIONAL,
                                      -- Timer unit: 5 seconds
          eventReportMask         [3] IMPLICIT BITSTRING OPTIONAL,
          lossOfSynchCounter      [4] IMPLICIT INTEGER OPTIONAL,
          loopTimeoutCounter      [5] IMPLICIT INTEGER OPTIONAL }
```

**Figure 67 - Parameters defined for each B Channel Layer 1 - Circuit mode**

To be defined in the future Standard

**Figure 68 - Parameters defined for each B Channel Layer 1 - Frame mode**

To be defined in the future Standard

**Figure 69 - Parameters defined for each B Channel Layer 2 - Frame mode**

To be defined in the future Standard

**Figure 70 - Parameters defined for each B Channel Layer 1 - Packet mode**

To be defined in the future Standard

**Figure 71 - Parameters defined for each B Channel Layer 2 - Packet mode**

To be defined in the future Standard

**Figure 72 - Parameters defined for each B Channel Layer 3 - Packet mode**

In the SEQUENCE case, the contents of the data element (i.e. the parameter Id) is equal to the tag of the same parameter of the IMPLICIT SET case. Only one example is given hereafter in Figure 73.

```
DefinedParameterId ::= INTEGER {
            .                       .
            .                       .
           N202                    (4),
            .                       .
            .                       . }

-- end of the "SEQUENCE" case
```

**Figure 73 - General Structure of the SEQUENCE Case**

```
ResourceStatusValue ::= INTEGER {
                            onlineIdle (1),
                            onlineBusy (2),
                            offline (3),
                            maintenance (4) }
```

**Figure 74 - ResourceStatusValue**

```
ActionType ::=     CHOICE {
           privateAction [5] ANY,
           definedAction [6] IMPLICIT DefinedAction }
```

**Figure 75 - ActionType**

```
DefinedAction ::= INTEGER {
                  LoopActiv (1),
                  LoopDeactiv (2),
                  LoopDeactivRq (3),
                  SelfTestRq (4),
                  SelfTestActiv (5),
                  SelfTestCanc (6),
                  SelfTestCont (7),
                  TestCall (8),
                  StartEventReport (9),
                  StopEventReport (10) }
```

**Figure 76 - DefinedAction**

```
ActionArgValue ::= CHOICE {
          privateArg    [0] ANY OPTIONAL,
          loopReference [1] IMPLICIT LoopReference OPTIONAL,
          loopTimer     [2] IMPLICIT INTEGER OPTIONAL,
                                    -- Timer unit: 5 seconds
          eventReportMask [3] IMPLICIT BITSTRING OPTIONAL }
```

**Figure 77 - ActionArgValue**

*Note 13*

*A bit set to ONE in the EventReportMask indicates that the event value corresponding to that bit position must be reported.*

*A bit set to ZERO in the EventReportMask indicates that the event value corresponding to that bit position must not be reported.*

*Bit positions are numbered starting from ZERO.*

*For example, to get event report only on loss of Synch (code: 3) EventReportMask should be equal to 8 (bit 3 = ONE).*

```
CompareOperat ::= INTEGER {
                  equal (2),
                  notEqual (3),
                  lessThan (4),
                  greaterThanOrEqual (5),
                  lessThanOrEqual (6),
                  greaterThan (7) }
```

**Figure 78 - CompareOperate**

```
LoopReference ::= INTEGER {
                    A5 (0),
                    A4 (1),
                    A3 (2),
                    A2 (3),
                    A1 (4),
                    B1 (5),
                    B2 (6),
                    B3 (7),
                    B4 (8) }
```

**Figure 79 - LoopReference**

## 9. USE OF LOWER LAYER SERVICES

This Technical Report allows the OSMAE to be supported by a full seven layer hierarchy or by a convergence function, allowing the OSMAE to directly use Layer 3 services.

The communication path between Layer 3 and the next higher layer is referenced by a Service Endpoint Identifier (SEI). In the outgoing direction the SEI is used as a reference at the Layer 3 boundary. In the incoming direction, the Service Access Point selector is extracted from the User-User information (UUI) element of the Layer 3 message and used to select the relevant protocol stack as depicted in Figure 80.

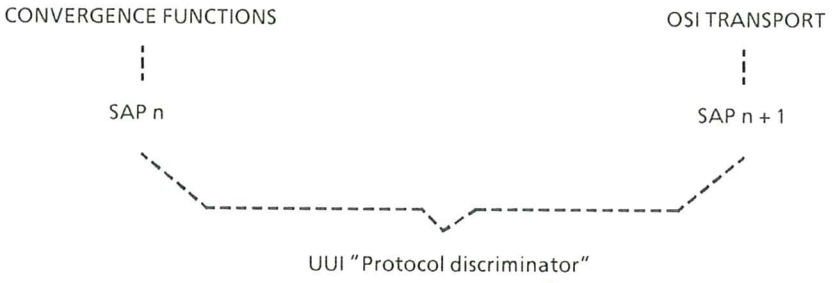



**Figure 80 - Selection of a Protocol Stack based on dedicated SAPs**

### 9.1 Full Seven Layer Hierarchy

The use of a full OSI hierarchy should be based on existing and specified OSI services.

Further study is necessary on the special requirements for the management context and the ISDN environment. Particular areas to be studied further are:

- unsupported features of ISDN interfaces: possible specification of the bearer channel and of its use by the application;
- non-support of connectionless services by the upper layers (under study at ISO).

Examples of the mappings that should take place when the lower services are defined are given hereafter.

### 9.1.1 Presentation Layer Services in the Connection-Oriented Mode

The OSMAE uses the Presentation Layer services via ACSE and ROSE. The establishment and release of an association is made via ACSE, while the actual data transfer is made via ROSE. This is shown in Tables 7 and 8.

| Function | ACSE | Presentation |
|---|---|---|
| Association Control | A-Association-RQ/IND | P-Connect-RQ/IND |
| | A-Association-RSP/CNF | P-Connect-RSP/CNF |
| | A-Release-RQ/IND | P-Release-RQ/IND |
| | A-Release-RSP/CNF | P-Release-RSP/CNF |
| | A-Abort-RQ/IND | P-U-Abort-RQ/IND |
| | A-P-Abort-/IND | P-P-Abort-IND |

**Table 7 - Use of Presentation Layer Services via ACSE**

| Function | ROSE | Presentation |
|---|---|---|
| Data Transfer | Invoke-RQ/IND | P-Data-RQ/IND |
| | Result-RQ/IND | P-Data-RQ/IND |
| | Error-RQ/IND | P-Data-RQ/IND |
| | Reject-RQ/IND | P-Data-RQ/IND |

**Table 8 - Use of Presentation Layer Services via ROSE**

### 9.1.2 Connectionless Presentation Layer Services in the Connectionless Mode

This subject needs further study.

## 9.2 Convergence Functions

The convergence functions will be reduced to a minimum and will allow the direct use of the Layer 3 services by mapping the ROSE and ACSE PDUs into the service elements specified in Standard ECMA-106.

Vice-versa, the OSMAE is directly accessed by Layer 3 via a specific SAP (see Figure 80). In this case the information required to identify the maintenance context to be carried in the APDU is limited to:

- the application context descriptor (name) which identifies the environment and the protocol in use;
- the (optional) user data which are provided by the OSMAP as part of the M-INITIALIZE primitive parameters. These are composed of:
  . access control information,
  . maintenance process identity.

### 9.2.1 Addressing

Depending on the location and addressability of the executor OSMAP, two variants may exist:

1. The executor's address exists: it is at a remote access or inside the PSN but is addressable. In this case the two modes (CO and CL) can be used and will make use of the addressing information. The Layer 3 user (OSMAE) data will be carried by the User-User Information element by invoking the primitives such as indicated in Tables 10 and 11.

2. The executor's address is unknown because the management function is implicit in the network. In this case the OSMAE will use a Layer 3 Connectionless mode due to the existence of the implicit permanent virtual circuit provided by the N-MGT-DATA-REQ/IND primitives. In this case the Maintenance Information element is used to carry the OSMAE data.

The convergence functions will therefore make the decision of mapping either of the primitives based on the existence of an addressing information: address provided in CL mode, address or connection reference in CO mode. This is depicted in Table 9.

| Layer 3 Service Elements | OSMAE Reqest Parameters | | |
|---|---|---|---|
| | Address | Layer 3 Mode | Section |
| N-xxx-yyy primitives | Yes<br>Yes | CO<br>CL | 9.2.2 |
| N-MGT-yyy primitives | No | CL | 9.2.3 |

**Table 9 - Mapping as a Function of the Addressing Mode**

### 9.2.2 Convergence Functions Conducted by Explicit Addressing

In the CO mode of operation the following features will be part of the convergence functions:

- Multiplexing of associations over a Layer 3 connection shall not be supported. As a consequence any association will be mapped into one Layer 3 connection and each connection can support no more than one association.
- Any request for activity on an unexisting Layer 3 connection shall be answered by a Layer 3 disconnect indication and translated into a negative response or an association abort.
- The associations shall not survive a lower layer disconnection and any unexpected Layer 3 connection release shall be mapped into an association abort.
- When an error primitive is used to indicate congestion in the PSN, it may also indicate that information has been lost. Such an error primitive will not be relayed by the convergence functions. The recovery mechanism of the OSMAP is expected to detect and recover from the loss of information.

When in the case of an already existing communication path the OSMAP knows that the maintenance activity to be performed pertains to an existing connection, it will use the connection's Service Endpoint Identifier (SEI) instead of the network address to indicate this fact.

Thus, the convergence functions shall use either:

network address — in a request for the establishment of a new Layer 3 connection, or

SEI — in a request for executing the maintenance activity on an already existing connection.

*Note 14*

*As an implementation option the Layer 3 call reference (CR) may be used instead of the SEI to uniquely identify an already existing connection. This choice is beyond the scope of this Technical Report.*

Tables 10 and 11 summarize the mappings within the convergence functions to be used in both situations.

| OSMAE ⇒ | Layer 3 Primitives and Type of Addressing | |
|---|---|---|
| | Special Maintenance Call (Network Address) | Existing Call (SEI) |
| A.Establishment Req | N-Connect-Req + Data ⇒ | N-Data-Req ⇒ |
| A.Establishment Rsp+ | N-Connect-Rsp + Data ⇒ | N-Data-Req ⇒ |
| A.Establishment Rsp- | N-Disconnect-Rsp + Data ⇒ | N-Data-Req ⇒ |
| A.Release Req | N-Disconnect-Req + Data ⇒ | N-Data-Req ⇒ |
| A.Abort Req | N-Disconnect-Req + Data ⇒ | N-Data-Req ⇒ |
| A.Release Rsp | N/A-Disconnect is unconfirmed *(Note 15)* | N-Data-Req ⇒ *(Note 16)* |
| Data Req | N-Data-Req ⇒ | N-Data-Req ⇒ |
| Unit-Data | N-Unit-Data-Req ⇒ | |

**Table 10 - Convergence Functions for Outgoing Activity**

*Note 15*

*When convergence functions are used, Release Req causes Layer 3 to disconnect. Since Release is not a user confirmed service at Layer 3 the A-Release-Response cannot be carried through. Release is therefore non-negotiable and must always be accepted. In the case indicated here the A-Release-Response primitive is simply ignored by Layer 3.*

*Note 16*

*In the case of a maintenance operation associated to an existing call the Layer 3 connection is owned by its initiator and the OSMAP cannot directly cause its release.*

| OSMAE ⇐ | Layer 3 Primitives and Type of Addressing | |
|---|---|---|
| | Special Maintenance Call (Network Address) | Existing Call (SEI) |
| A.Establishment Ind | ⇐ N-Connect-Ind + Data | ⇐ N-Data-Ind |
| A.Establishment Cnf+ | ⇐ N-Connect-Cnf + Data | ⇐ N-Data-Ind |
| A.Establishment Cnf- | ⇐ N-Disconnect-Cnf + Data | ⇐ N-Data-Ind |
| A.Release Ind | ⇐ N-Disconnect-Ind + Data | ⇐ N-Data-Ind |
| A.Abort Ind | ⇐ N-Disconnect-Ind + Data | ⇐ N-Data-Ind |
| A.Release Cnf | ⇐ N-Disconnect-Cnf *(Note 17)* | ⇐ N-Data-Ind |
| A-P-Abort | ⇐ N-Disconnect-Ind *(Note 18)* | ⇐ N-Data-Ind |
| Data Ind | ⇐ N-Data-Ind | ⇐ N-Data-Ind |
| Unit-Data | ⇐ N-Unit-Data-Ind | |

**Table 11 - Convergence Functions for Incoming Activity**

*Note 17*

*The N-Disconnect does not carry any data. Convergence functions must generate an A-Release-Cnf PDU. The alternative is to consider the A-Release as an unconfirmed service but this would introduce an incompatibility with its use in a full OSI stack.*

*Note 18*

*The N-Disconnect carries no user data and the cause returned indicates a network problem.*

### 9.2.3 Automatic Routing to the Network Management Centre

When the end system does not know or does not want to know the address of the OSMAP because it is located within the PSN, a special maintenance message will be generated outside any call by invoking the N-MGT-DATA-REQ primitive. This message is routed based on its semantics and the service offered can be considered as based on a permanent virtual circuit. In this sense it can be considered as a CL service with an implicit destination address. The source address needs to be carried somehow to allow the network OSMAP to respond.

The mapping will then be done as shown in Table 12.

| Network Service | ACSE | ROSE |
|---|---|---|
| N-MGT-DATA | A-<prim> | RO-<prim> |

**Table 12 - Mapping for Implicit Addressing**

# APPENDIX A

## IMPACT ON STANDARD ECMA-106

Although beyond the scope of this Technical Report, some indications are given in this Appendix on the way Standard ECMA-106 supports maintenance activities.

This should allow the reader to better understand the overall context.

### A.1 Indication of the Maintenance Context

The maintenance context is indicated at Layer 3 by a special indicator (maintenance flag) that allows the execution of actions specific to a maintenance call (bypass barred accesses, authorize abnormal use of a given access like calling itself, ...).

The Layer 3 entity will know that the maintenance context indicator should be set when the primitives will be issued over a dedicated SAP.

The Service Endpoint Identifier will have a similar structure to the Layer 2 CEI:

$$(SEI)3 = SAPI + CES$$

where in the CO mode the Connection Endpoint Suffix (CES) will allow the Layer 3 user to identify a particular logical channel to the Layer 3 service provider. In the CL mode the CES is irrelevant and will be ignored.

### A.2 Layer Management Activity

Any maintenance activity relevant to layer management is carried by special Layer 3 information elements. Examples of activities which are considered as layer management are:

- changing status of an access/channel,
- establishing Layer 3 logical loop.

### A.3 Application Selection

The Layer 3 management shall carry information that allows the selection of the maintenance application in a transparent way (e.g. protocol discriminator in the U-U information element).

### A.4 Support of a Fast Select Mechanism

The Layer 3 elements and procedures shall include a Fast Select-like mechanism allowing a connectionless service to be offered to the Layer 3 user.

### A.5 PSN/ISDN Layer 3 Service Elements

The service elements offered by the Layer 3 protocol are summarized hereafter. They are not described in the present Edition of Standard ECMA-106.

N-CONNECT (REQUEST, INDICATION, RESPONSE, CONFIRM): This service is used to control the PSN connections with or without B-channels.

N-DATA (REQUEST, INDICATION): This service is used to pass user-to-user information down to and up from the signalling channel.

N-DISCONNECT (REQUEST, INDICATION, RESPONSE, CONFIRM): This service is used to control the tear-down of connections established across the PSN.

N-FACILITY (REQUEST, INDICATION): This service is used to control PSN facilities. Presently, it is not supported by Standard ECMA-106.

N-UNIT-DATA (REQUEST, INDICATION): This service is used to access a network employing the connectionless mode of operation. Presently, it is not supported by Standard ECMA-106.

N-MGT-DATA (REQUEST, INDICATION): This service is used to exchange information between OSMAPs (one of these being located in the network maintenance centre) using implicit addressing.

Table A.1 summarizes the various primitive parameters.

| Layer 3 Primitives | Parameters | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | SEI *(Note 3)* | Mainte-nance Indication | Destina-tion Address | Source Address *(Note 1)* | Cause | User Data | Bearer Service Descrip-tion *(Note 2)* | Layer 3 CR |
| N-CONNECT REQ. | X | X | X | O | | X | X/D | |
| N-CONNECT IND. | X | | | P | | X | X | X |
| N-CONNECT RSP | X | | | | | X | O | |
| N-CONNECT CNF. | X | | | | | X | P | X |
| N-DISCONNECT REQ. | X | | | | X/D | X | | |
| N-DISCONNECT IND. | X | | | | X | X | | |
| N-DISCONNECT CNF. | X | | | | | | | |
| N-Data REQ. | X | | | | | X | | O |
| N-Data IND. | X | | | | | X | | O |
| N-UNIT-DATA REQ. | *(Note 3)* | X | X | X | | X | | O |
| N-UNIT-DATA IND. | *(Note 3)* | | | X | | X | | |
| N-MGT-DATA REQ. | *(Note 3)* | | X *(Note 4)* | X | | X | | O *(Note 5)* |
| N-MGT-DATA IND. | *(Note 3)* | | | X *(Note 4)* | | X | | O *(Note 5)* |

Legend: X = Applies O = Optional P = Possible, if option is used. X/D = Parameter may be a default.

**Table A.1 - Layer 3 Primitives and their Parameters**

*Note A.1*

*The source address may not be known by the requestor, in which case it will ask the Layer 3 entity to.use the calling address supplementary service.*

*Note A.2*

*Bearer service description includes service and QoS definitions which will be used to construct the BC and LLC information elements. The content of this (set of) parameter(s) requires further study.*

*Note A.3*

*The SEI contains the SAPI. It will be used in both CO and CL modes to instruct Layer 3 to set the maintenance indicator flag when necessary (in unit-data and in connect primitives). As a consequence the maintenance indicator is not explicitly passed down.*

*Note A.4*

*Implicit or ignored at the maintained end system side.*

*Note A.5*

*The use of the CR of an existing call is for further study.*

# APPENDIX B

## DEFINITIONS OF ROSE MACROS AND APDUs

### B.1 Remote Operations Data Type Definitions

The notation of the APDUs interchanged between OSMAEs follows the rules set out in X.ROS0 (see ISO DIS 9072/1). Examples for ROSE data types are given in this Appendix.

```
Remote-Operation-Notation (joint-iso-ccitt remote Operations (4) notation (0)
DEFINITIONS::=

BEGIN

EXPORTS BIND, UNBIND, OPERATION, ERROR;
-- macro definition for bind-operations

BIND MACRO::=

BEGIN
TYPE NOTATION                   ::=Argument Result Error
VALUE NOTATION                  ::=value (VALUE NULL)
Argument                        ::= "ARGUMENT" Named Type | empty
Result                          ::= "RESULT" Named Type | empty
Error                           ::= "BIND-ERROR" Named Type | empty
Named Type                      ::=identifier type | type
END

-- macro definition for unbind-operations

UNBIND MACRO::=

BEGIN
TYPE NOTATION                   ::=Argument Result Error
VALUE NOTATION                  ::=value (VALUE NULL)
Argument                        ::= "ARGUMENT" Named Type | empty
Result                          ::="RESULT" Named Type | empty
Errors                          ::="UNBIND-ERROR" Named Type | empty
Named Type                      ::=identifier type | type
END

-- macro definition for operation

OPERATION MACRO ::=
BEGIN
```

```
TYPE NOTATION          ::=Argument Result Errors Linked Operations
VALUE NOTATION         ::=value (VALUE CHOICE  {
                                        localValue INTEGER,
                                    globalValueSEQUENCE {OBJECT IDENTIFIER,INTEGER}})
Argument               ::="ARGUMENT" NamedType | empty
Result                 ::="RESULT" NamedType | empty
Result Type            ::=NamedType | empty
Errors                 ::= "ERRORS"  "{"ErrorNames"}" | empty
LinkedOperations       ::="LINKED" "{"LinkedOperationNames "}" | empty
ErrorNames             ::=ErrorList | empty
ErrorList              ::=Error |  ErrorList ","Error
Error                  ::=value (ERROR)
LinkedOperationNames   ::=OperationList | empty
OperationList          ::= Operation | OperationList "," Operation
Operation              ::=value (OPERATION)
NamedType              ::=identifier type | type
END


-- macro definition for operations errors

ERROR MACRO::=
BEGIN

TYPE NOTATION          ::=Parameter
VALUE NOTATION         ::=value (VALUE CHOICE  {
                                        localValue INTEGER,
                                    globalValueSEQUENCE {OBJECT IDENTIFIER,INTEGER}})
Parameter              ::= "PARAMETER" NamedType | empty
NamedType              ::=identifier type | type
END


END      -- end of Remote Operations Notation



Remote-Operation-Notation-extension {(joint-iso-ccitt remote Operations (4) notation-
extension (2)}
DEFINITIONS::=

BEGIN

EXPORTS APPLICATION-SERVICE-ELEMENT,APPLICATION-CONTEXT;
IMPORTS                        OPERATION,BIND,UNBIND FROM      RemoteOperation-Notation
                                                               {joint-iso-ccitt
                                                               remoteOperations(4)notation
                                                               (0);
-- macro definition for Appl.Service Element (ASE)

APPLICATION-SERVICE-ELEMENT MACRO ::=
BEGIN

TYPE NOTATION              ::=SymmetricAse | Consumerinvokes Supplierinvokes | empty
VALUE NOTATION             ::=value (VALUE OBJECT IDENTIFIER)
SymmetricAse               ::="OPERATIONS" "{"OperationList"}"
Consumerinvokes            ::="CONSUMER INVOKES" "{"OperationList"}" | empty   (not used)
Supplierinvokes            ::="SUPPLIER INVOKES" "{"OperationList"}" | empty   (not used)
OperationList              ::=Operation | OperationList ","Operation
Operation                  ::=value (OPERATION)
END
```

```
-- macro definition for application-contexts
APPLICATION-CONTEXT MACRO::=
BEGIN

TYPE NOTATION              ::=NoneROelements Binding ROelements AbstractSyntaxes
VALUE NOTATION             ::=value (VALUE OBJECT IDENTIFIER)
NonROelements              ::="APPLICATION SERVICE ELEMENTS" "{" AseList "}"
Binding                    ::="BIND" value (BIND)
                              "UNBIND" value (UNBIND)
ROelements                 ::="REMOTE OPERATIONS" "{"AseID "}"   -- identifying ROSE
                              SymmetricAses Asymmetric Ases | empty
SymmetricAses              ::="OPERATIONS OF" "{"AseList "}" | empty
AsymmetricAses             ::=InitiatorConsumerOf ResponderConsumerOf          (not used)
InitiatorConsumerOf        ::="INITIATOR CONSUMER OF" "{"AseList"}" | empty    (not used)
ResponderConsumerOf        ::="RESPONDER CONSUMER OF" "{"AseList"}" | empty    (not used)
AbstratSyntaxes            ::="ABSTRACT SYNTAXES" "{"AbstractSyntaxList"}"
AseList                    ::=AseID | AseList "," AseID
AseID                      :=value (APPLICATION-SERVICE-ELEMENT)
AbstractSyntaxList         ::=AbstractSyntax | AbstractSyntaxList "," AbstractSyntax
AbstractSyntax             ::=value (OBJECT ITENTIFIER)    -- identifying abstract syntax

END

END          -- end of Remote Operations Notation extension
```

## B.2 Remote Operations Protocol Definitions

The protocol syntax for the interchange of APDUs follows the rules set out in X.ROS1 (see ISO DIS 9072/2).

```
-- Remote-Operations-APDUs {joint-iso-ccitt remoteOperations(4) apdus(1)}

DEFINITONS :: =
BEGIN
EXPORTS rOSE, invokeID Type;

-- the following macros are used as defined in Figure 4 of ISO 9072/1 /

IMPORTS OPERATION, ERROR FROM Remote-Operation-Notation {joint-iso-ccitt remoteOperations(4) notation(0)};

rOSE OBJECT IDENTIFIER    :: = {joint-iso-ccitt remoteOperations(4) aseID(3)}

-- APDUs

ROSEapdu                  :: = CHOICE{ [1]IMPLICIT ROIV apdu,
                                       [2]IMPLICIT RORS apdu,
                                       [3]IMPLICIT ROER apdu,
                                       [4]IMPLICIT RORJ apdu}

-- APDU types

ROIVapdu                  :: = SEQUENCE{
                                    invokeID invokeIDType,
                                    linked-ID(0) | IMPLICIT INTEGER OPTIONAL,
                                    operation-value OPERATION,
                                    argument ANY OPTIONAL}
invokeIDType              :: = INTEGER

RORSapdu                  :: = SEQUENCE {
                                    invokeID invokeIDType,
                                    result ANY OPTIONAL}
ROERapdu                  :: = SEQUENCE {
                                    invokeID invokeIDType,
                                    error-value ERROR,
                                    parameter ANY OPTIONAL}
RORJapdu                  :: = SEQUENCE {
                                    invokeID CHOICE {invokeIDType, NULL}
                                    problem CHOICE {
                                       [0]IMPLICIT GeneralProblem,
                                       [1]IMPLICIT InvokeProblem,
                                       [2]IMPLICIT ReturnResultProblem,
                                       [3]IMPLICIT ReturnErrorProblem}}
```

```
-- APDU Rejection Reason
GeneralProblem      ::= INTEGER{                    -- ROSE-provider detected
                        unrecognizedAPDU (0),
                        mistypedAPDU (1),
                        badlyStructuredAPDU (2)}
InvokeProblem       ::= INTEGER{                    -- ROSE-user detected
                        duplicateinvocation (0),
                        unrecognizedOperation (1),
                        mistypedArgument (2),
                        resourceLimitation (3),
                        initiatorReleasing (4),
                        unrecognizedLinkedID (5),
                        linkedResponseUnexpected (6),
                        unexpectedChildOperation (7)}
ReturnResultProblem ::= INTEGER{                    -- ROSE-user detected
                        unrecognizedinvocation (0),
                        reultResponseUnexpected (1),
                        mistypedResult (2)}
ReturnErrorProblem  ::= INTEGER{                    -- ROSE-user detected
                        unrecognizedinvocation (0),
                        errorResponseUnexpected (1),
                        unrecognizedError (2),
                        unexpectedError (3),
                        mistypedParameter (4)}

END                 -- end of ROSE Protocol specification
```

# APPENDIX C

## PRIMITIVE MAPPING USING CONNECTION-ORIENTED SERVICES

The following diagrams show the mapping of the various primitives for application association establishment, application association release and management operation exchange. Only the use of the Layer 3 connection-oriented services is shown. Parts 1, 3 and 5 describe the use of convergence functions while Parts 2, 4 and 6 describe the use of a full OSI stack.

**Part 1 Association of an Application - Convergence Protocol**

Figure C-1 shows the successful establishment of an application association.

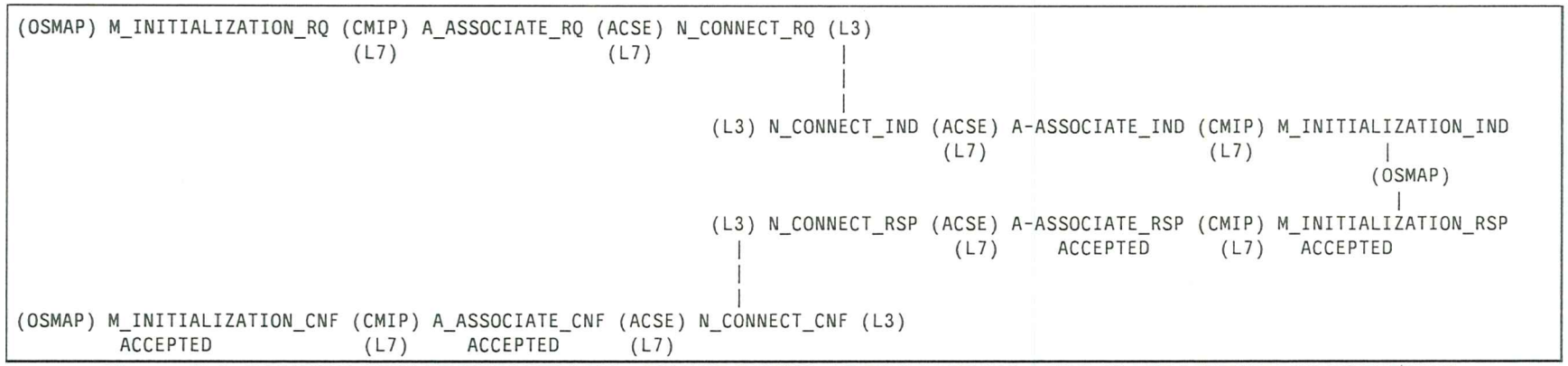



**Figure C-1: CONVERGENCE FUNCTIONS - SUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT**

If the Layer 3 connection exists already at the time of application association establishment, the N__DATA primitive will be used instead of the N__CONNECT primitive.

In Figure C-2, the application association is unsuccessful due to rejection by the remote OSMAP.

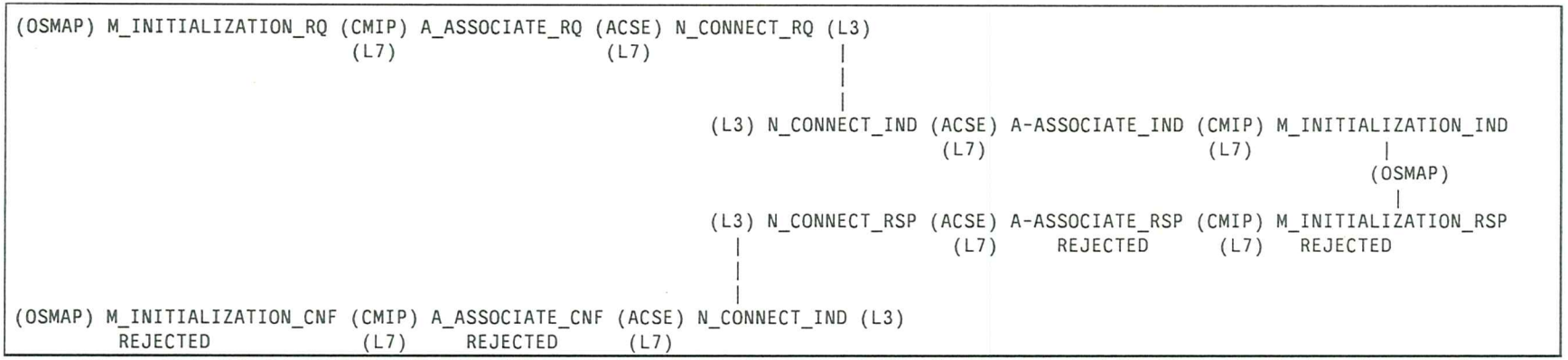



**Figure C-2: CONVERGENCE FUNCTIONS - UNSUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT**
**Association rejected by the OSMAP**

In Figure C-3, the application association is unsuccessful due to rejection by the remote CMIP.

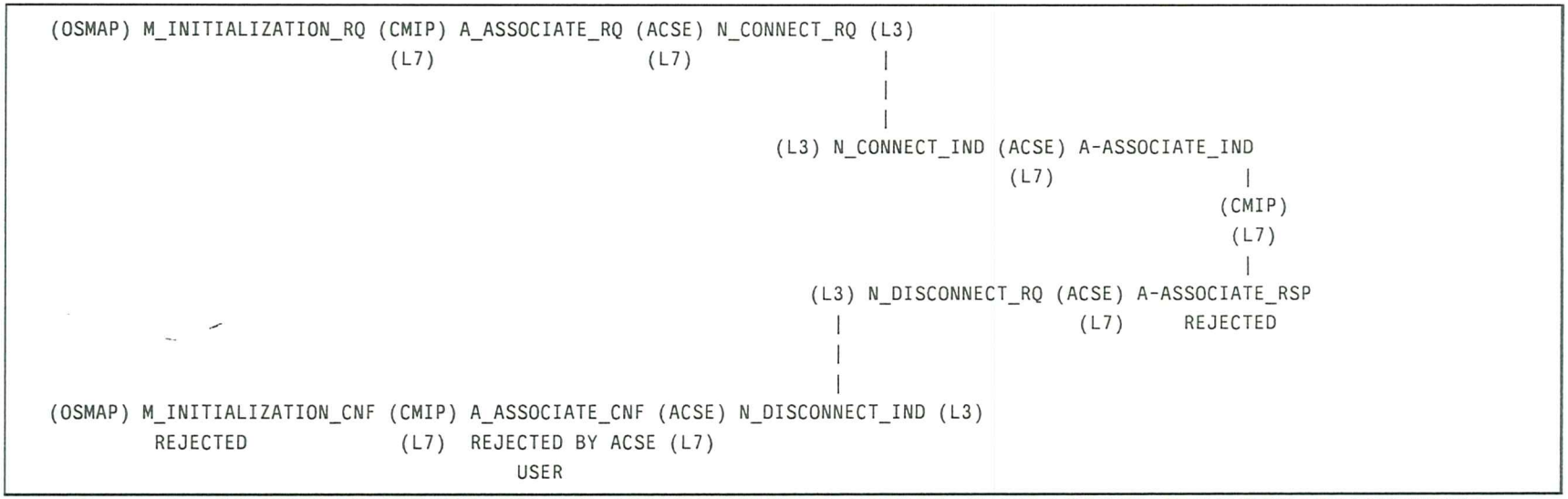



**Figure C-3: CONVERGENCE FUNCTIONS - UNSUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT**
**Rejection by CMIP (L7)**

In Figure C-4, the application association establishment is unsuccessful due to rejection by the remote ACSE.

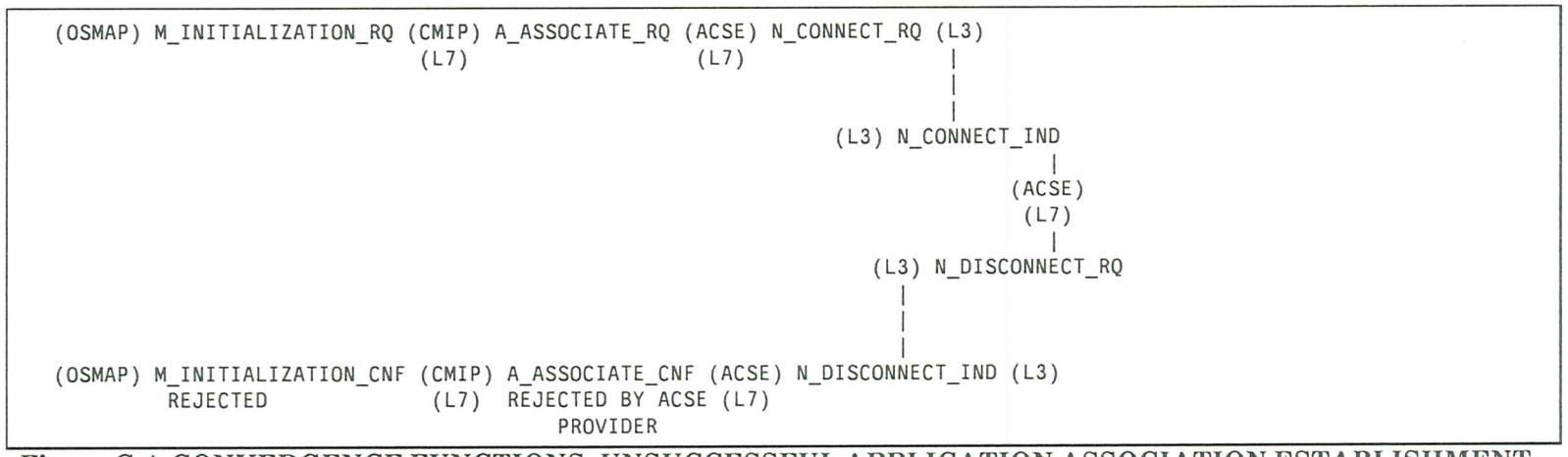



**Figure C-4: CONVERGENCE FUNCTIONS- UNSUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT Rejection by ACSE (L7)**

In Figure C-5, the application association establishment is unsuccessful due to rejection by the local network service provider.

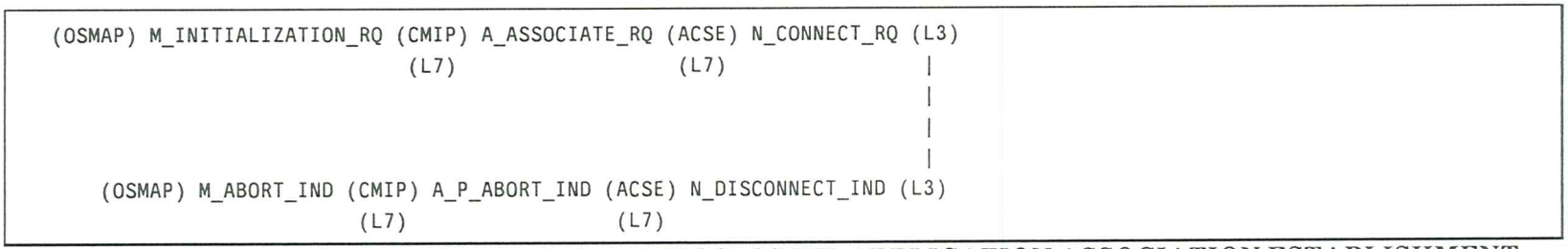



**Figure C-5: CONVERGENCE FUNCTIONS - UNSUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT Association rejected by the Local Network Service Provider (L3)**

In Figure C-6, the application association establishment is unsuccessful due to rejection by the remote network service provider.

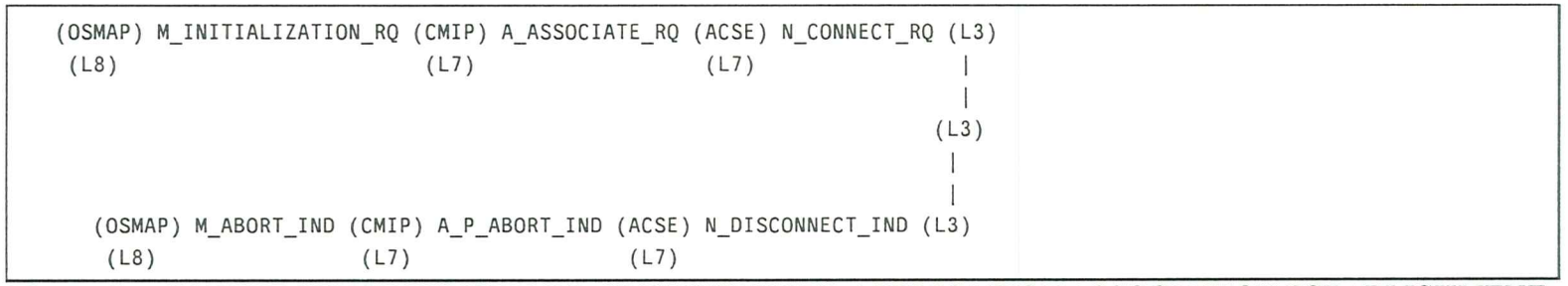



**Figure C-6: CONVERGENCE FUNCTIONS - UNSUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT**
**Association rejected by the Remote Network Service Provider (L3)**

Figures C-7 to C-9 show the primitive parameters exchanged at the local SMI (OSMAP ↔ ACSE), at the SMDSI (ACSE ↔ Layer 3) and at the remote SMI (OSMAP ↔ ACSE).

| **M_INITIALIZE_RQ/IND Paramaters** | **CMIP** | **A_ASSOCIATE_RQ/IND Parameters** |
|---|---|---|
| Called OSMAP Address | <<----------->> | Recipient Application Entity Title |
| Not used | <<----------->> | Initiator Application Entity Title |
| Not used | <<----------->> | Quality of Service |
| | "ECMA-CMIP" --------->> | Application Context Name |
| Access Control | <<----------->> | User Information |
| Maintenance Proicess Number | <<----------->> | User Information |
| N O T U S E D | | Session Requirements |
| N O T U S E D | | Calling Presentation Address |
| N O T U S E D | | Called Presentation Address |
| N O T U S E D | | Session Connection Identifier |
| N O T U S E D | | Presentation Context Definition List |
| N O T U S E D | | Default P-Context Name |
| N O T U S E D | | Presentation Requirements |
| N O T U S E D | | Initial Synch Point Serial Number |
| N O T U S E D | | Initial Assignments of Tokens |

**Figure C-7: CONVERGENCE FUNCTIONS - PRIMITIVE PARAMETERS OSMAP ↔ ACSE**

| A_ASSOCIATE_RQ/IND Parameters | ACSE | N_CONNECT RQ/IND Parameters |
|---|---|---|
| Recipient Application Entity Title <<----->> | L7 Directory Function <<------->> | Called Network Address |
| | For Convergence | |
| | AARQ APDU <<--------------->> | User Information |
| N O T U S E D | | Quality of Service |
| N O T U S E D | | Requested Bearer Service (Default: B channel) |
| N O T U S E D | | Requested B-Channel (Default: any B channel) |

**Figure C-8: CONVERGENCE FUNCTIONS - PRIMITIVE PARAMETERS ACSE ↔ Layer 3**

| M_INITIALIZE_RQ/IND Parameters | CMIP | A_ASSOCIATE_RQ/CNF Parameters |
|---|---|---|
| Called OSMAP Address | <<-------------------------->> | Recipient Application Entity Title |
| Result | <<-------------------------->> | Result |
| | "ECMA-CMIP" -------->> | Application Context Name |
| Not used | <<-------------------------->> | Quality of Service |
| Not used | <<-------------------------->> | Session Requirements |
| Not used | <<-------------------------->> | Session Connection Identifier |
| Not used | <<-------------------------->> | Responding Presentation Address |
| Not used | <<-------------------------->> | Presentation Requirements |
| Not used | <<-------------------------->> | Initial Synch Point Serial Number |
| Not used | <<-------------------------->> | Initial Assignments of Tokens |

**Figure C-9: CONVERGENCE FUNCTIONS - PRIMITIVE PARAMETERS OSMAP ↔ ACSE**

The A__ASSOCIATE__RSP/CNF primitive is mapped into the N__CONNECT__RSP/CNF primitive or into the N__DISCONNECT__RQ/IND primitive depending on the value of the Result parameter of the A__ASSOCIATE. If the A__ASSOCIATE Result parameter is "accepted" the A__ASSOCIATE__RSP/CNF is mapped into the N__CONNECT__RSP/CNF. If the A__ASSOCIATE Result parameter is "rejected" the A__ASSOCIATE__RSP/CNF is mapped into the N__DISCONNECT__RQ/IND. This is shown in Figures C-10 and C-11.

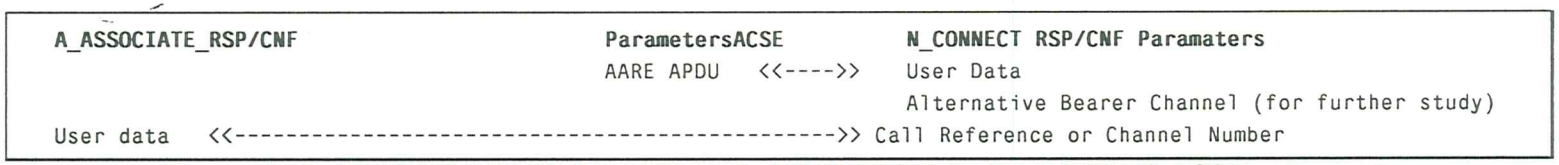



**Figure C-10: CONVERGENCE FUNCTIONS - PRIMITIVE PARAMETERS ACSE ↔ Layer 3 Association Request Accepted**

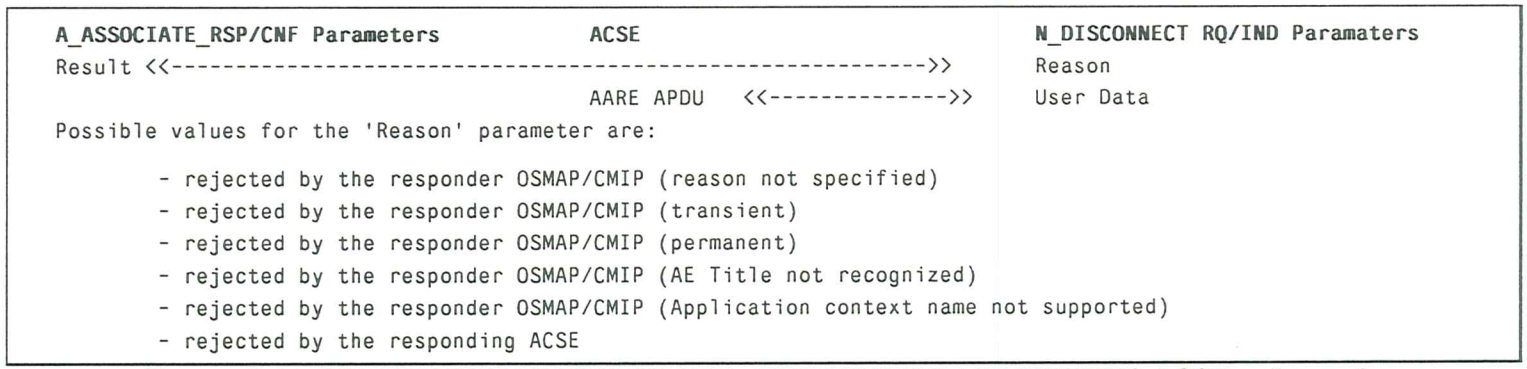



**Figure C-11: CONVERGENCE FUNCTIONS - PRIMITIVE PARAMETERS ACSE ↔ Layer 3 Association Request Rejected**

**Part 2 Full OSI Stack (All Seven Layers are Available)**

Figure C-12 shows the successful establishment of an application association.

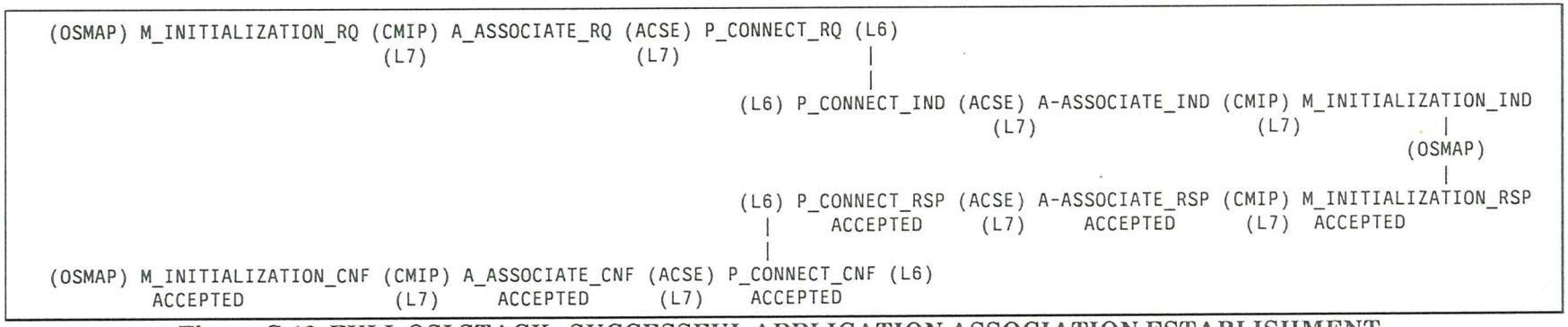



**Figure C-12: FULL OSI STACK - SUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT**

Figure C-13 shows the unsuccessful application association establishment, due to rejection by the OSMAP.

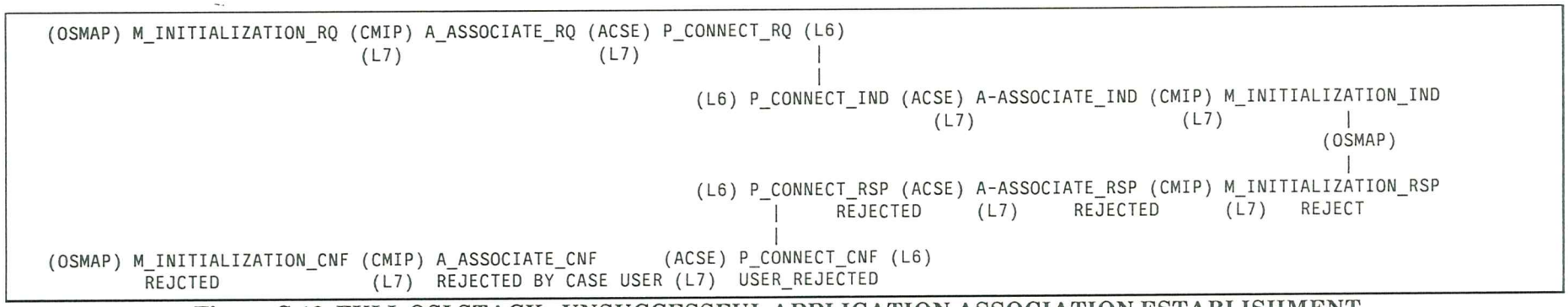



**Figure C-13: FULL OSI STACK - UNSUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT**
**Rejection by the OSMAP**

Figure C-14 shows the unsuccessful application association establishment, due to rejection by the CMIP.

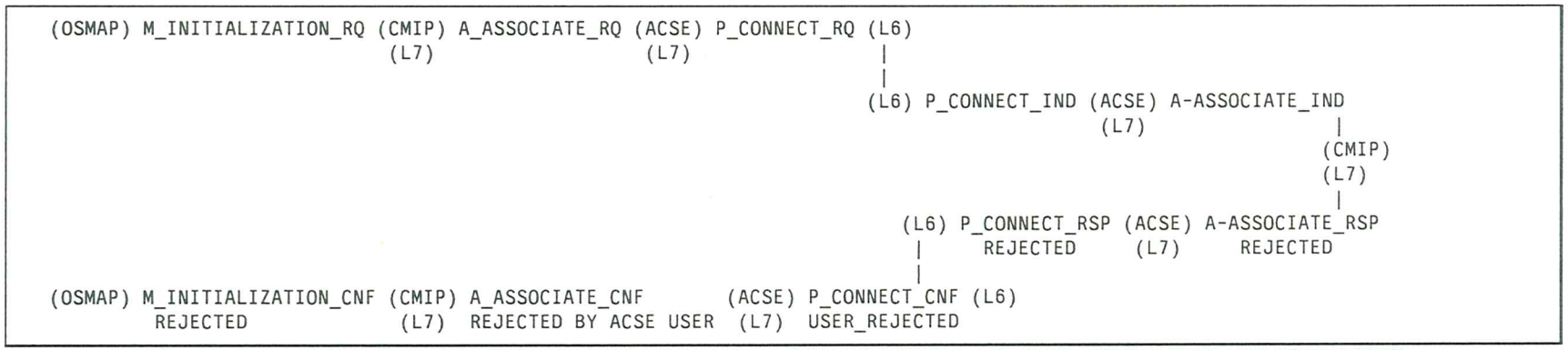



**Figure C-14: FULL OSI STACK - UNSUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT Rejection by the CMIP (L7)**

Figure C-15 shows the unsuccessful application association establishment, due to rejection by the ACSE.

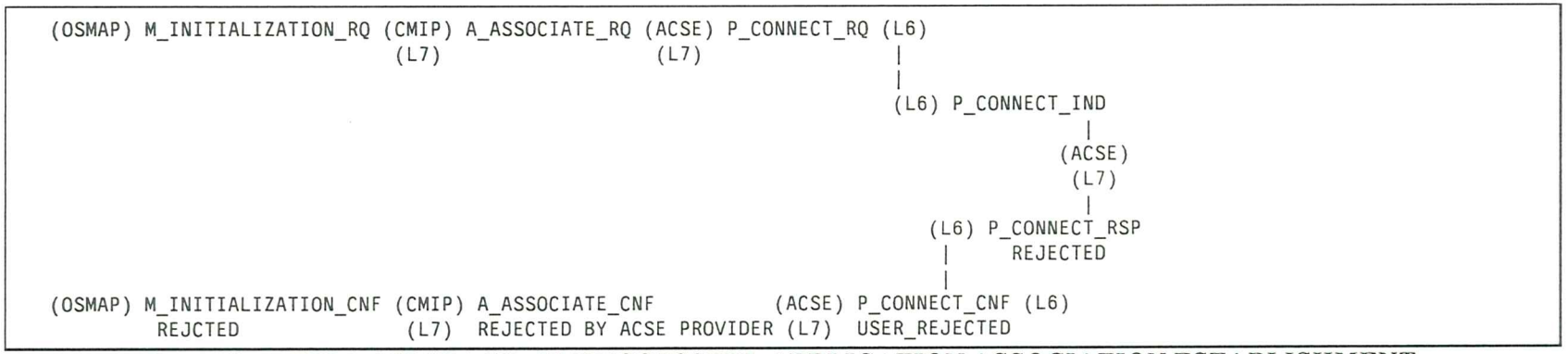



**Figure C-15: FULL OSI STACK - UNSUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT Rejection by the Local ACSE (L7)**

Figure C-16 shows the unsuccessful application association establishment, due to rejection by the local presentation service provider.

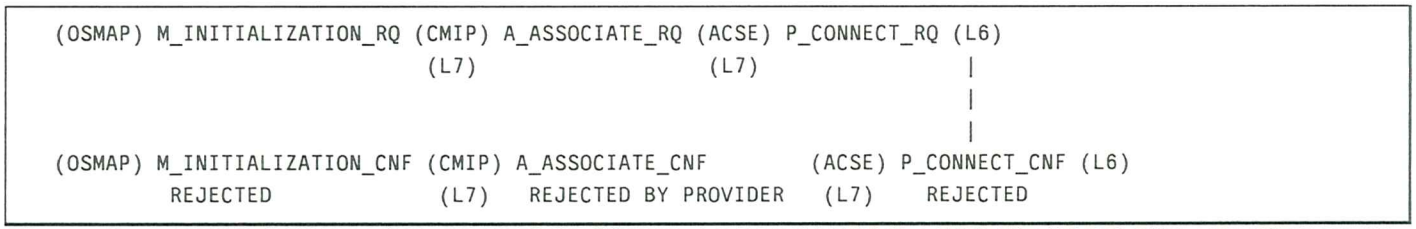



**Figure C-16: FULL OSI STACK - UNSUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT Rejection by the Local Presentation Service Provider (L6)**

Figure C-17 shows the unsuccessful application association establishment, due to rejection by the remote presentation service provider.

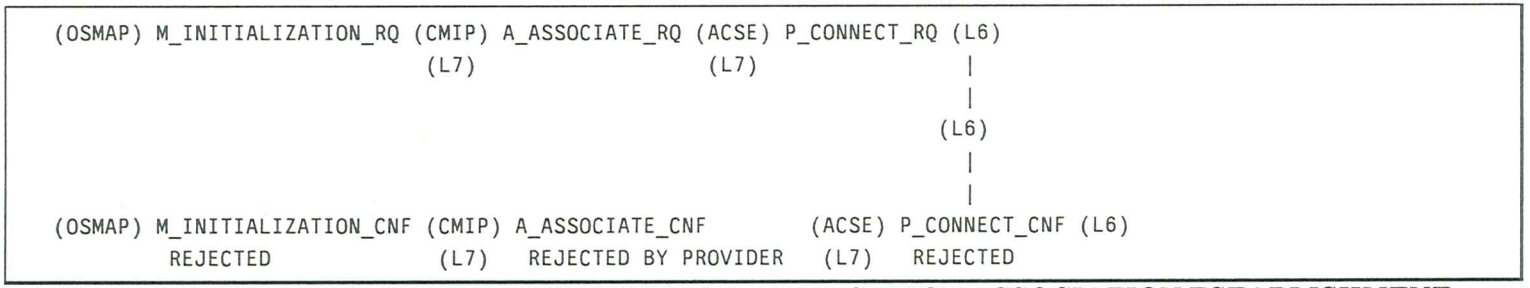



**Figure C-17: FULL OSI STACK - UNSUCCESSFUL APPLICATION ASSOCIATION ESTABLISHMENT Rejection by the Remote Presentation Service Provider (L6)**

Figures C-18 to C-21 show the primitive parameters to be interchanged at the local SMI (OSMAP ↔ ACSE), at the local SMDSI (ACSE ↔ Layer 6), at the remote SMI (OSMAP ↔ ACSE) and at the remote SMDSI (ACSE ↔ Layer 6).

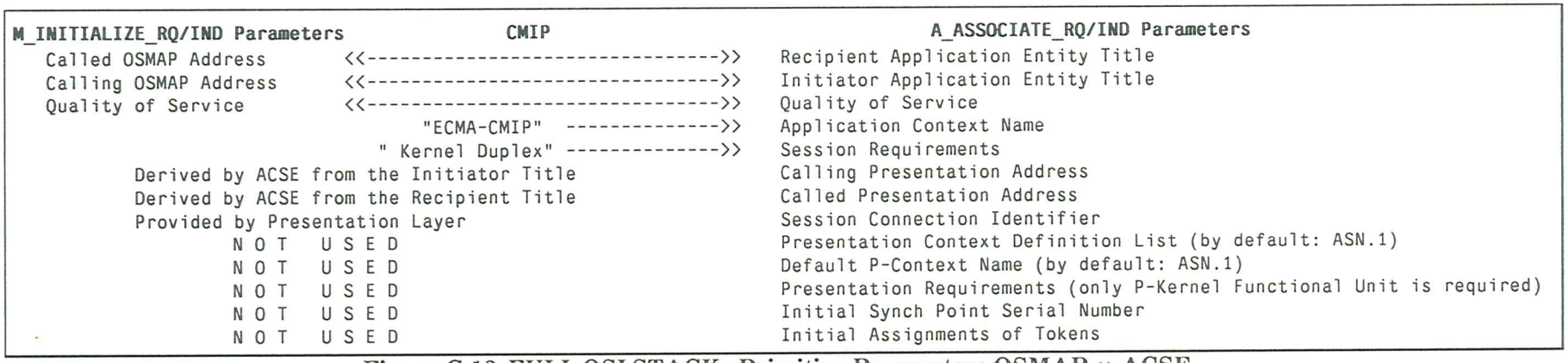



**Figure C-18: FULL OSI STACK - Primitive Parameters OSMAP ↔ ACSE**

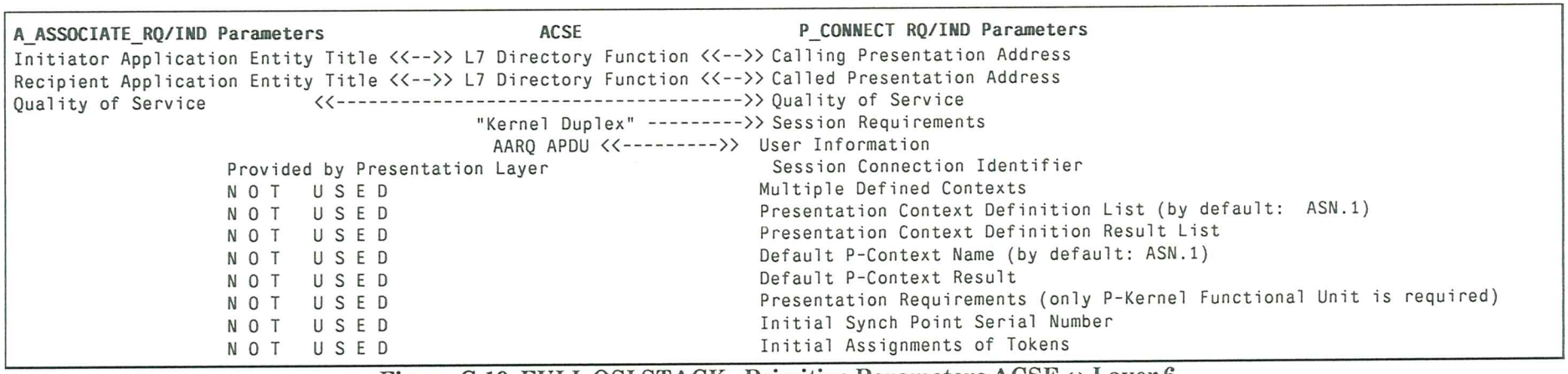



**Figure C-19: FULL OSI STACK - Primitive Parameters ACSE ↔ Layer 6**

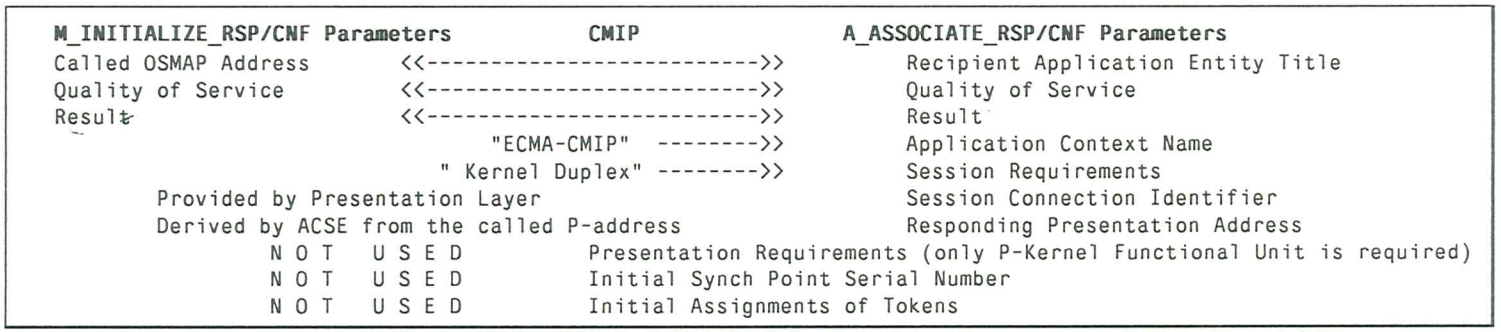



**Figure C-20: FULL OSI STACK - Primitive Parameters OSMAP ↔ ACSE**

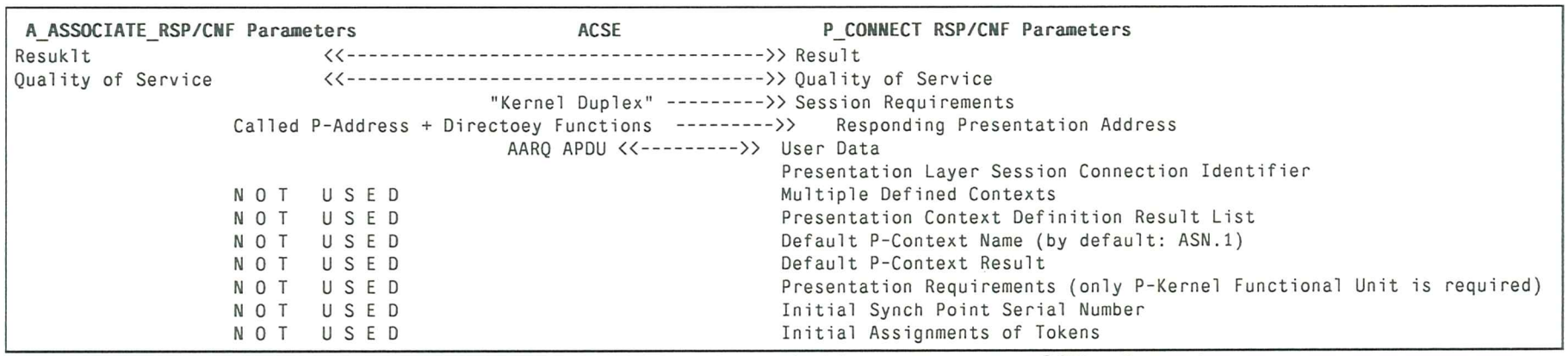



**Figure C-21: FULL OSI STACK - Primitive Parameters ACSE ↔ Layer 6**

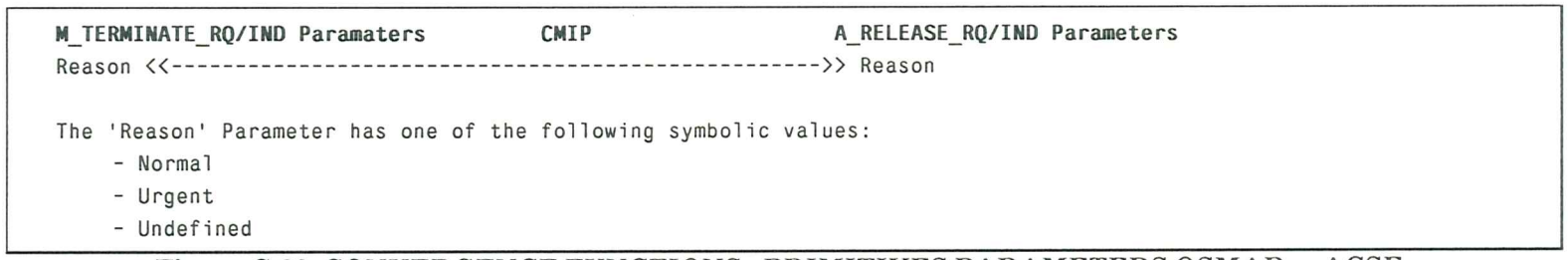



**Figure C-23: CONVERGENCE FUNCTIONS - PRIMITIVES PARAMETERS OSMAP ↔ ACSE**

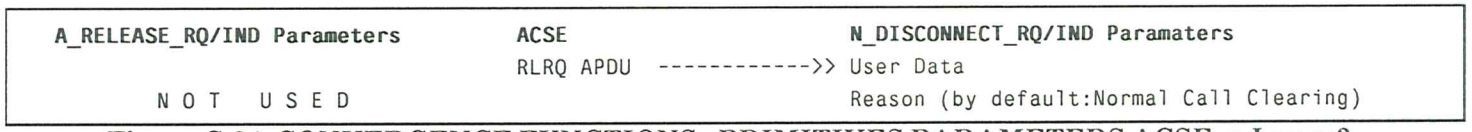



**Figure C-24: CONVERGENCE FUNCTIONS - PRIMITIVES PARAMETERS ACSE ↔ Layer 3**

In the case of a network problem, Layer 3 generates an N__DISCONNECT__IND with no user data and with a specific reason code. ACSE maps the N__DISCONNECT__IND into the A__P__ABORT__IND .

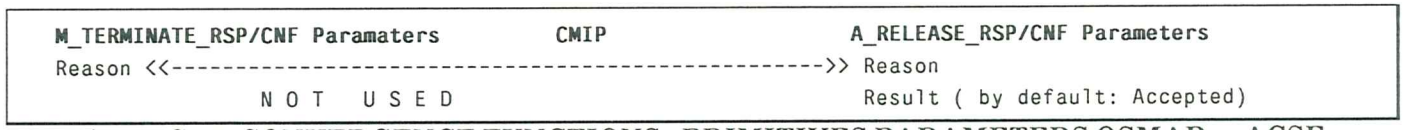



**Figure C-25: CONVERGENCE FUNCTIONS - PRIMITIVES PARAMETERS OSMAP ↔ ACSE**

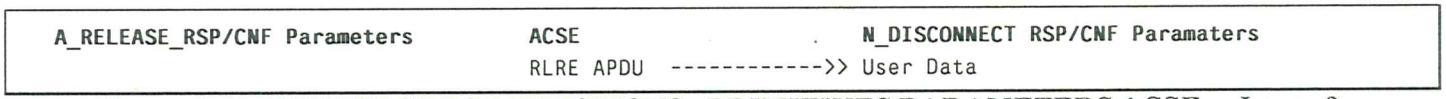



**Figure C-26: CONVERGENCE FUNCTIONS - PRIMITIVES PARAMETERS ACSE ↔ Layer 3**

**Part 3 Association Release - Convergence Functions**

If the Layer 3 connection already exists before and independently of the application association (i.e. the N__DATA primitive was used to map the A__ASSOCIATE__RQ), the N__DATA primitive will also be used to map the A__RELEASE__RQ (instead of the N__DISCONNECT).

According to ACSE, the responding user is permitted to reject the A__RELEASE request only if the Session Negotiated Release functional unit was selected for the association. This is never the case for CMIP. Therefore an A__RELEASE request can never be rejected by an ACSE user. Moreover, ACSE never rejects the Release request by its own decision, ie. it always generates an A__RELEASE__IND to its user. Therefore, an A__RELEASE request can never be rejected by ACSE.

Figure C-22 shows the sequence of a normal association release, while Figures C-23 to C-26 indicate the parameters to be exchanged at the local and remote SMIs and SMDSIs.

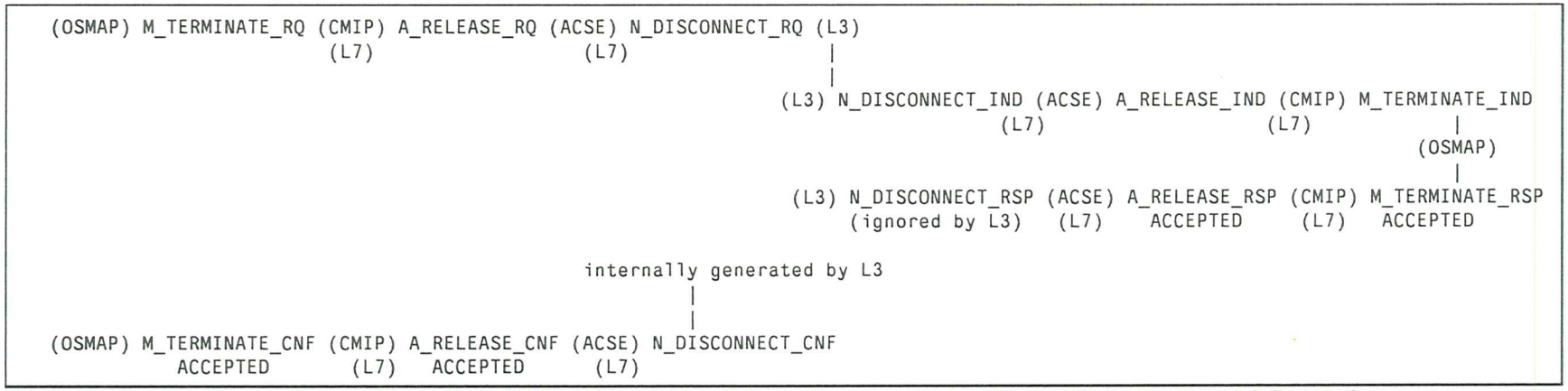



**Figure C-22: CONVERGENCE FUNCTIONS - NORMAL RELEASE OF AN ASSOCIATION**

### Part 4 Association Release - Full OSI Stack

The mapping as currently discussed in ISO will apply. This is shown in Figure C-27, while Figures C-28 to C-31 indicate the parameters to be exchanged at the SMIs and SMDSIs.

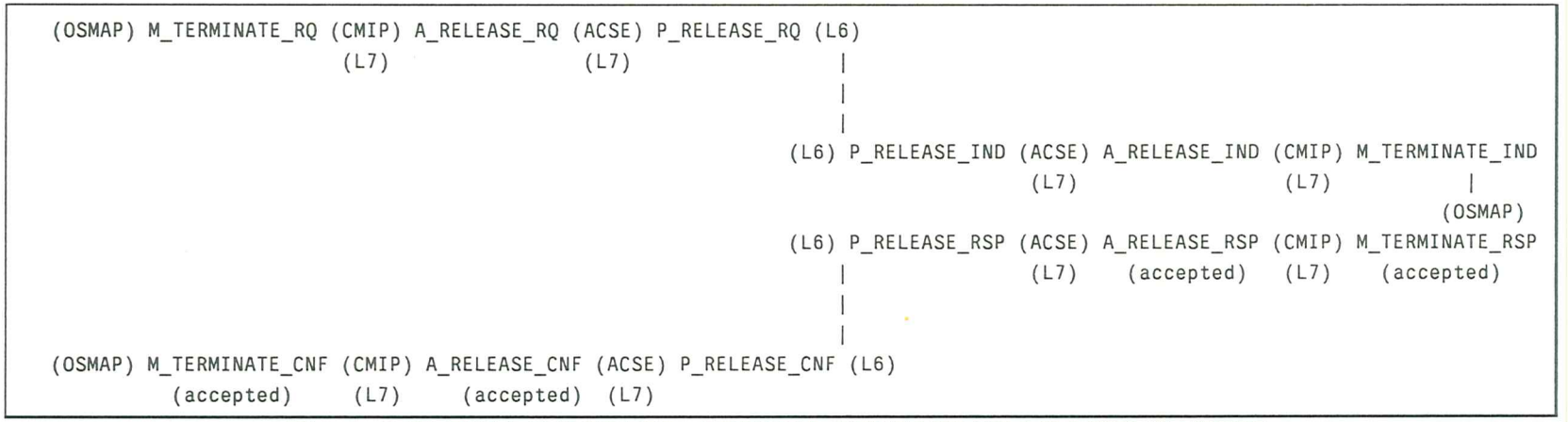



**Figure C-27: FULL OSI STACK - NORMAL RELEASE OF AN ASSOCIATION**

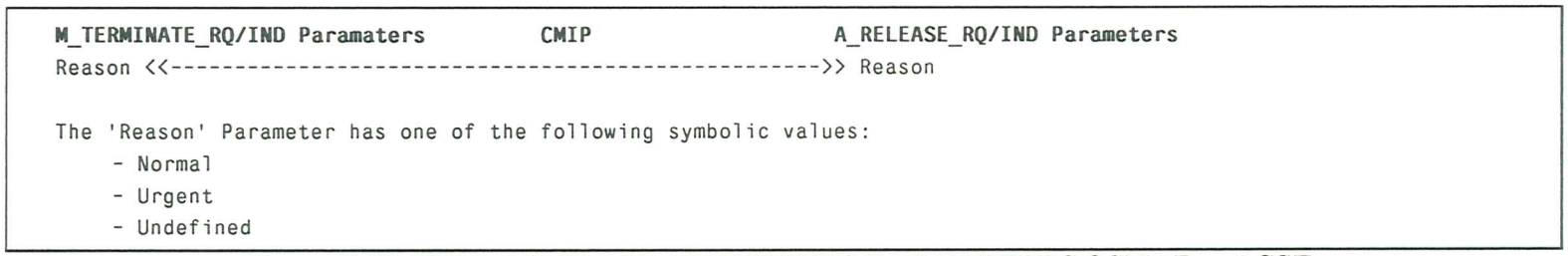



**Figure C-28: FULL OSI STACK- PRIMITIVES PARAMETERS OSMAP ↔ ACSE**

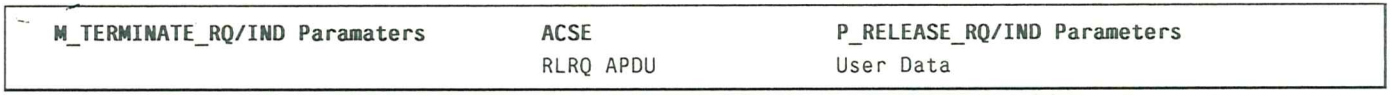



**Figure C-29: FULL OSI STACK- PRIMITIVES PARAMETERS ACSE ↔ Layer 6**

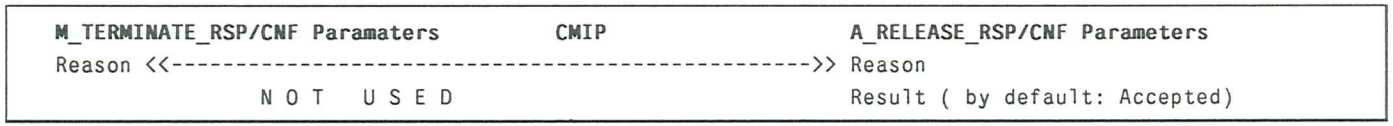



**Figure C-30: FULL OSI STACK- PRIMITIVES PARAMETERS OSMAP ↔ ACSE**

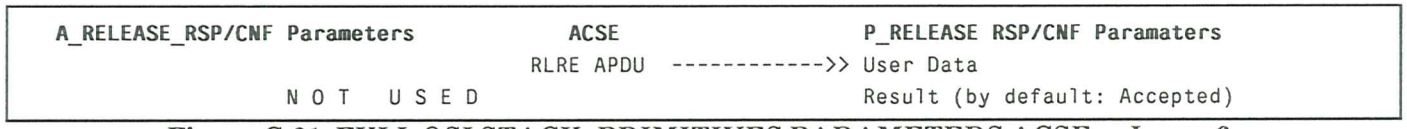



**Figure C-31: FULL OSI STACK- PRIMITIVES PARAMETERS ACSE ↔ Layer 6**

**Part 5 Management Operation - Convergence Functions**

Once the association is established, the Management Operations can be executed, ie. SET, GET, ACTION, BLOCKING, COMPARE, and CONFIRMED EVENT REPORT. Successful operations are shown in Figure C-32. For the unconfirmed EVENT REPORT the lower part of Figure C-32 does not apply.

The N__UNIT__DATA or the N__MGT__DATA primitives are used instead of the N__DATA primitive if Layer 3 connectionless service or implicit routing are selected.

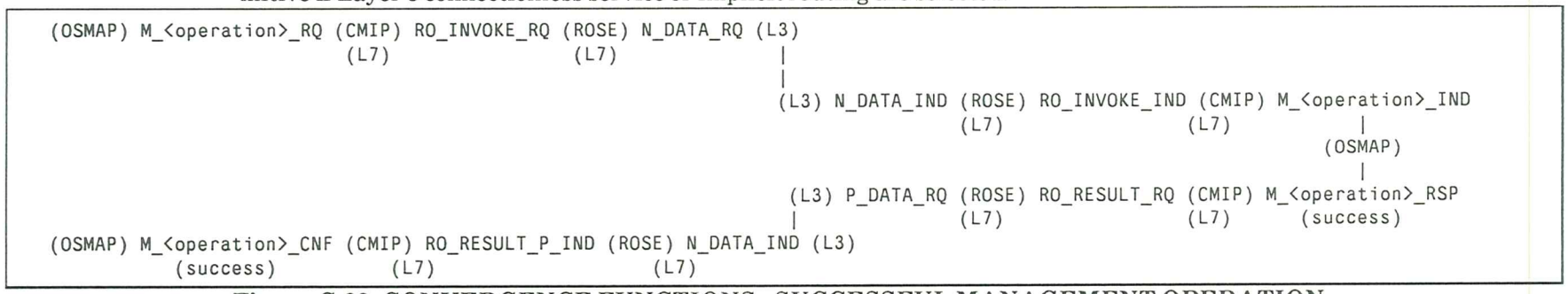



**Figure C-32: CONVERGENCE FUNCTIONS - SUCCESSFUL MANAGEMENT OPERATION**

Unsuccessful operations are shown in Figures C-33 (OSMAP Error), C-33 (OSMAP Rejection) and C-34 (Remote Rejection by ROSE).

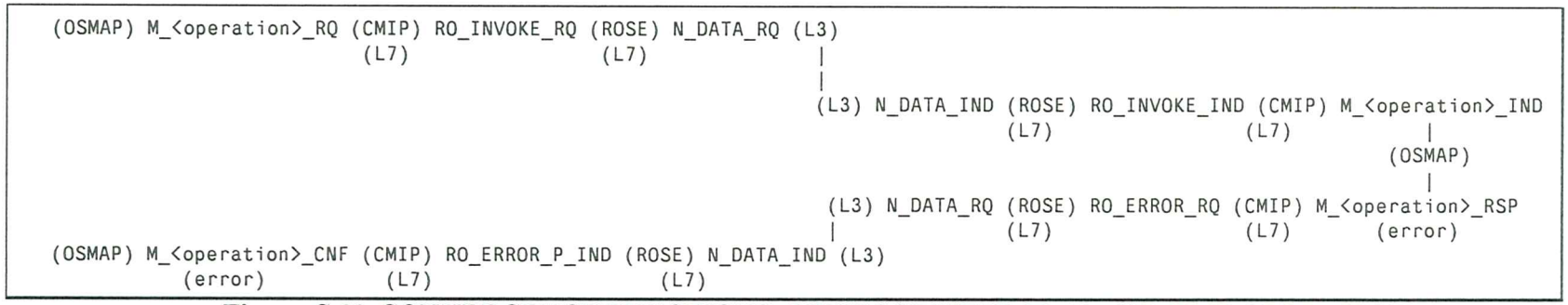



**Figure C-33: CONVERGENCE FUNCTIONS - UNSUCCESSFUL MANAGEMENT OPERATION OSMAP Error**

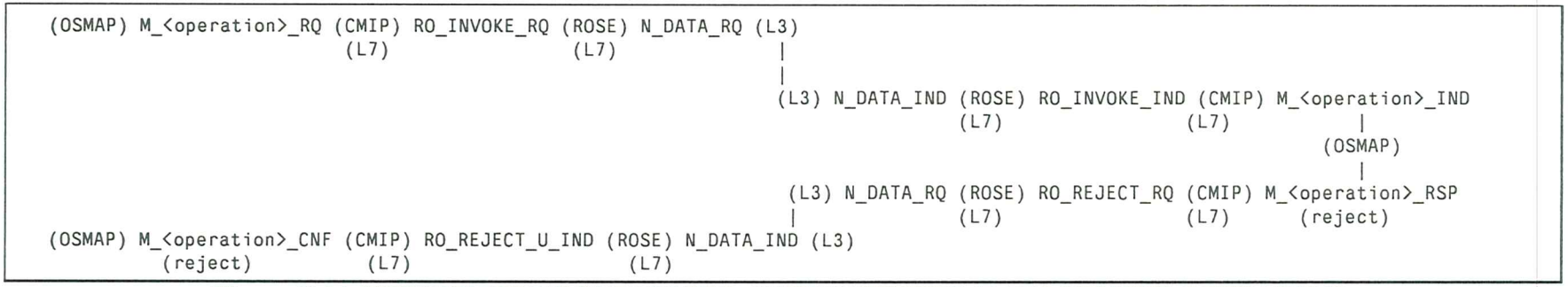



**Figure C-34: CONVERGENCE FUNCTIONS - UNSUCCESSFUL MANAGEMENT OPERATION OSMAP Rejection**

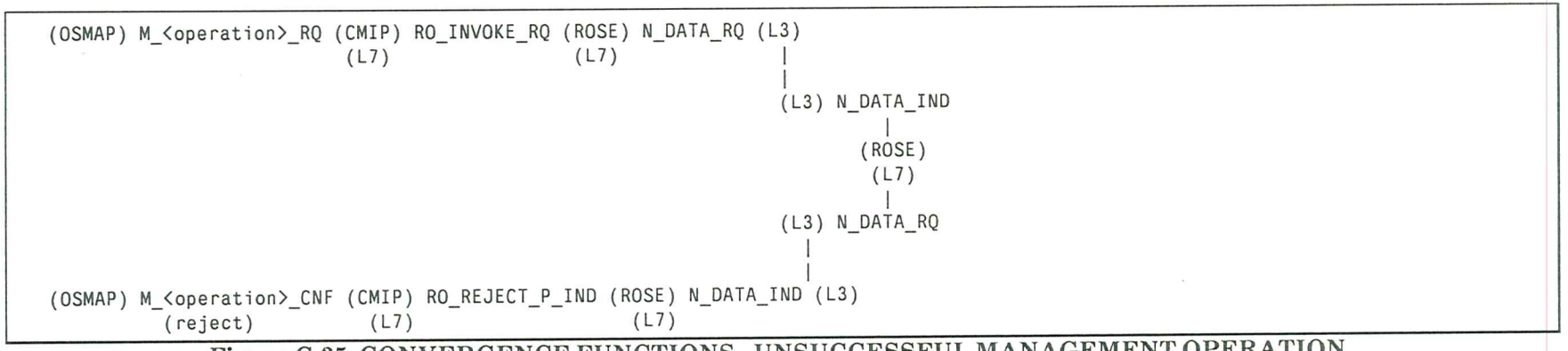



**Figure C-35: CONVERGENCE FUNCTIONS - UNSUCCESSFUL MANAGEMENT OPERATION**
**Remote Rejection by ROSE**

**Part 6 Management Operation - Full OSI Stack**

Once the association is established, the Management Operations can be executed, ie. SET, GET, ACTION, BLOCKING, COMPARE, and CONFIRMED EVENT REPORT. Successful operations are shown in Figure C-36. For the unconfirmed EVENT REPORT the lower part of Figure C-36 does not apply.

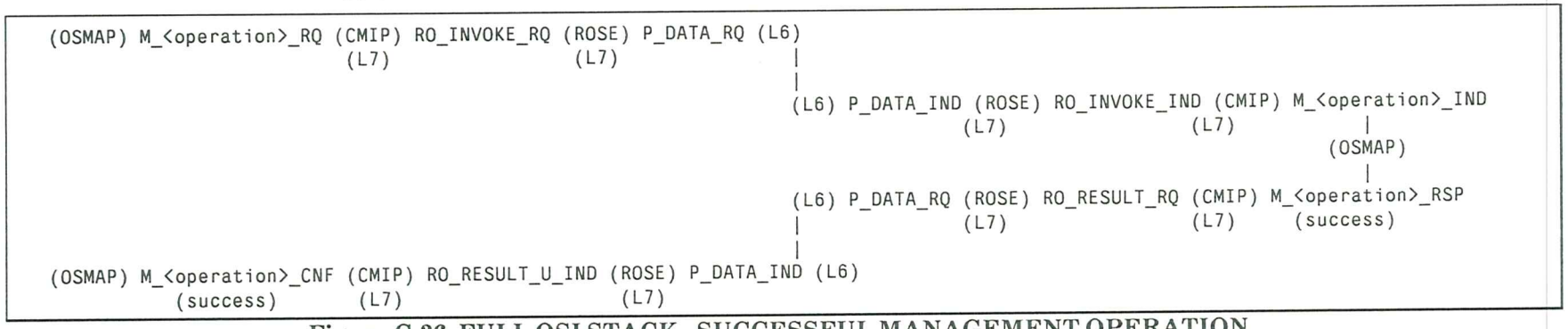



**Figure C-36: FULL OSI STACK - SUCCESSFUL MANAGEMENT OPERATION**

Unsuccessful operations are shown in Figures C-37 (OSMAP Error), C-38 (OSMAP Rejection), C-39 (Remote Rejection by ROSE) and C-40 (Local Rejection by ROSE).

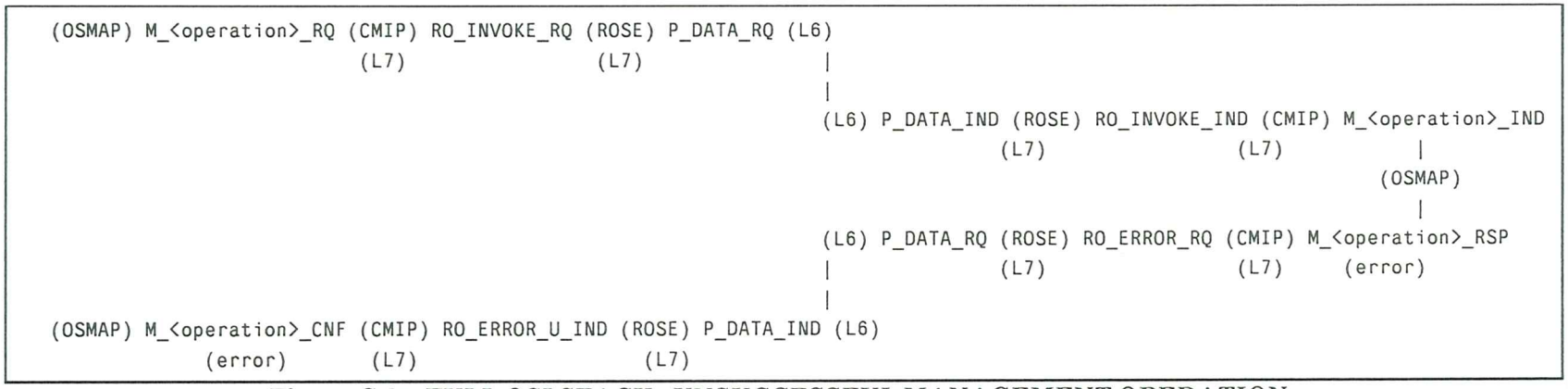



**Figure C-37: FULL OSI STACK - UNSUCCESSFUL MANAGEMENT OPERATION OSMAP Error**

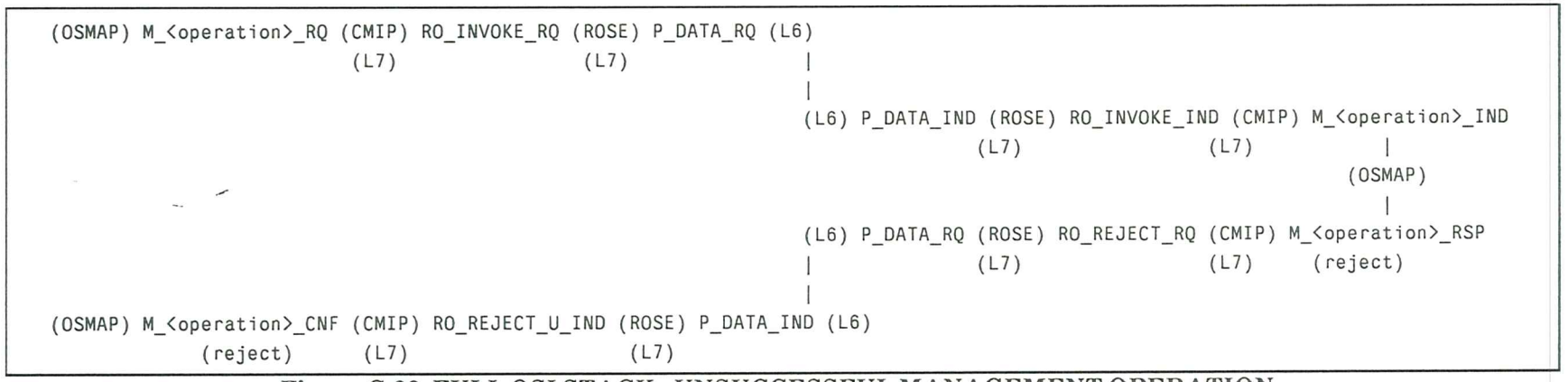



**Figure C-38: FULL OSI STACK - UNSUCCESSFUL MANAGEMENT OPERATION OSMAP Rejection**

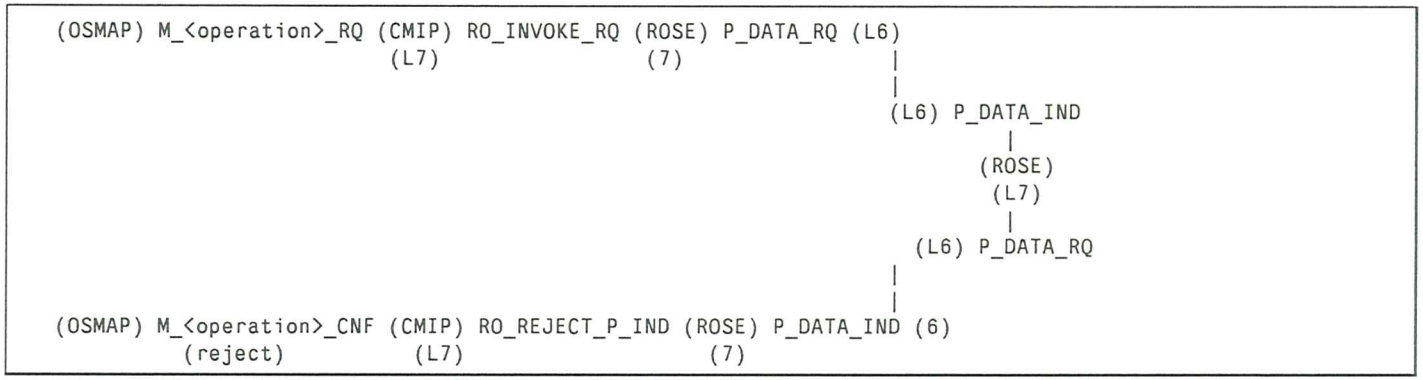



**Figure C-39: FULL OSI STACK - UNSUCCESSFUL MANAGEMENT OPERATION ROSE Remote Rejection**

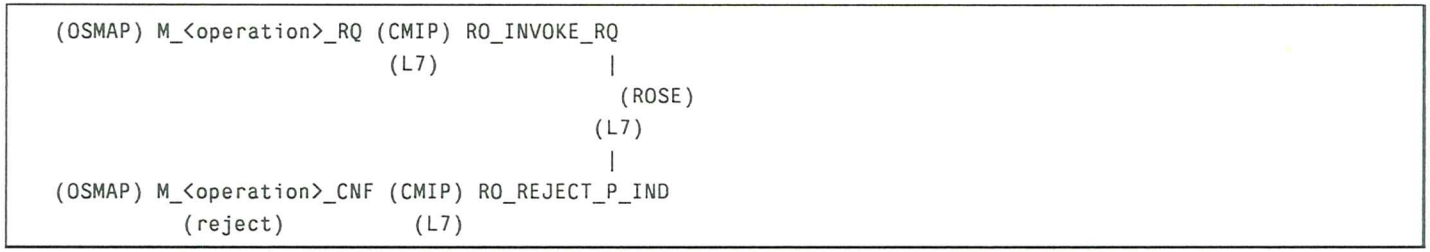



**Figure C-40: FULL OSI STACK - UNSUCCESFULL MANAGEMENT OPERATION ROSE Local Rejection**

# APPENDIX D

# PROTOCOL SYNTAX DESCRIPTION

The protocol specified by this Technical Report is based upon the transfer syntax and encoding defined in the ISO ASN.1 syntax and encoding standards. This Appendix describes briefly that syntax and notation.

## D.1 Data Types and Standard Notation

ASN.1 defines a transfer syntax for various kinds of information. Each piece of information is considered to have a type as well as a value. A data type is a class of information (for example, numeric or textual). A data value is an instance of such a class (for example, a particular number of a fragment of text). ASN.1 defines a number of generally useful data types, from which application-specific data types are constructed in this Technical Report, and others which make use of the ASN.1 transfer syntax. Among the generally useful data types defined by ASN.1 are Integer, Octet String, Sequence and Set.

The standard notation defined in ASN.1 is a formal description method that allows data types relevant for an application to be specified in terms of other data types, including generally useful data types of ASN.1. This notation is used in this Technical Report, where the protocol element data types, management definitions, and common definitions are specified in terms of others, and finally in terms of basic data types such as Integer and Octet String.

## D.2 Standard Representation of Data Types

The standard representation for a data type is the set of rules for encoding values of that type for transmission as a sequence of octets. The representation of a value also encodes its type and length, and is completely implied by the standard notation of the data type.

The standard representation of a data value is a data element having three components, which always appear in the following order:

- the Identifier designates the data type and governs the interpretation of the Contents.
- the Length specifies the length of the Contents,
- the Contents is the value of the data element, encoded as specified in ASN.1.

The Identifier and the Length each consist of one or more octets while the Contents consists of zero or more octets.

### D.2.1 Identifier

Four classes of data types are distinguished by means of the Identifier:

- universal,
- application-wide,
- context-specific, and
- private-use.

Universal types are generally useful, application-independent types; they are defined in ASN.1.

Application-wide types are more specialized, being peculiar to a particular application; they are defined in this Technical Report and in others using ASN.1 by means of the standard notation.

Context-specific types, like application-wide types, are peculiar to an application and are defined using the standard notation. However, they are used only within an even more limited context.

Private-use types are reserved for private use; the assignment of specific private-use identifiers can be accomplished by means of the standard notation but is outside the scope both of ASN.1 and of this Technical Report.

Two forms of data elements are distinguished by means of the Identifier: primitive and constructor. A primitive element is one of the Contents which is atomic. A constructor element is one of the Contents which is itself a data element or a series of data elements. Constructor elements are thus recursively defined.

### D.2.2 Length

The Length specifies the length (L) in octets of the Contents and is itself variable in length. It may take any of three forms: short, long and indefinite.

- the short form is used when L is less than 128,
- the long form is used when L is greater than 127,
- the indefinite form may (but need not) be used when the element is a constructor; when this form is employed, a special end-of-contents (EoC) element terminates the Contents.

### D.2.3 Contents

The Contents is variable in length and is interpreted in a type-dependent way. If the data element is a constructor, the contents itself comprises zero or more data elements; data elements are thus recursively defined.

## D.3 Built-in Types and Defined Types

The generally useful data types defined in ASN.1 consist of built-in types and defined types.

Built-in types are used to construct all other data types. They include Integer, Octet String, Sequence, Set and Tagged. Integer is a primitive data type. Octet String can be either primitive or constructor. Sequence and Set are constructor data types. Identifiers for these data types are of the universal class and are specified in ASN.1. A tagged data type is a data type for which the Identifier can be specified using the standard notation, as is done in the Definition Clause of this Technical Report.

Defined types are specified in ASN.1 using the standard notation. They include String and Printable String, which are defined in terms of the built-in type Octet String. They can be either primitive or constructor; their identifiers are of the universal class and are specified in ASN.1

## D.4 Box Representations

The APDU Identifier is coded as shown in Figure D.1.

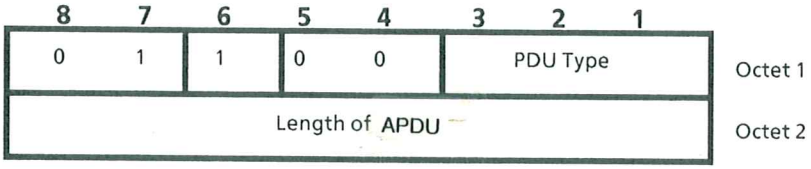



**Figure D.1 - APDU Identifier**

The INVOKE Identifier is coded as shown in Figure D.2.

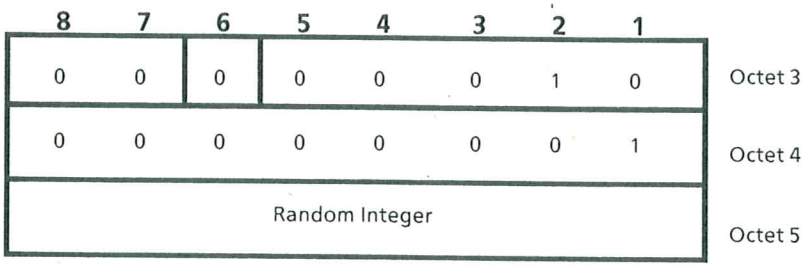



**Figure D.2 - INVOKE Identifier**

The OPERATION Code is coded as shown in Figure D.3.

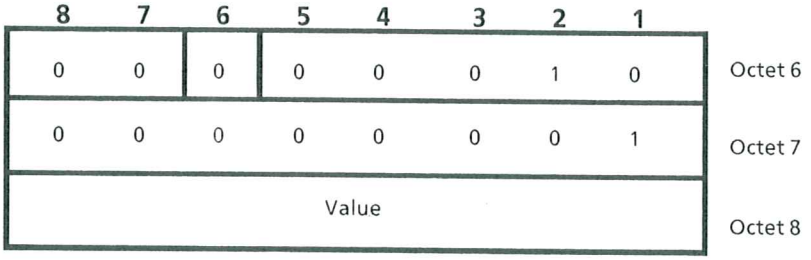



**Figure D.3 - OPERATION Code**

The PARAMETER LIST is coded as shown in Figure D.4.

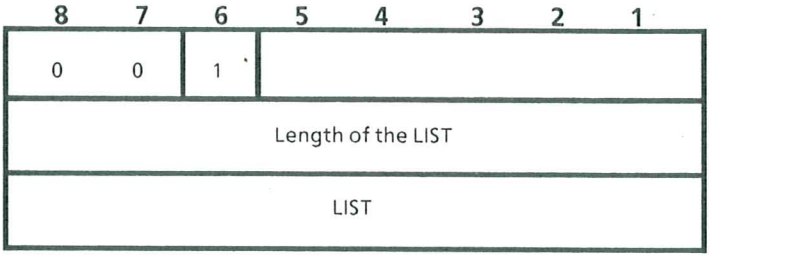



**Figure D.4 - PARAMETER LIST**

# APPENDIX E

## DETAILED CODING EXAMPLES

Some detailed coding examples are given hereafter to provide a clear view of the APDU in the case of ACTION and SET operations.

### E.1 Example for ACTION (Test Call involving a B Channel)

| | | |
|---|---|---|
| byte 1 | A1 | Class: Context - Constructor - Invoke |
| byte 2 | 16 | Length: 22 |
| byte 3 | 30 | SEQUENCE: Universal, Constructor Tag = 16. |
| byte 4 | 14 | Length: 20 bytes |
| byte 5 | 02 | INTEGER |
| byte 6 | 01 | Length: 1 byte |
| byte 7 | InvokeID | range 0 - 255 |
| byte 8 | 02 | INTEGER |
| byte 9 | 01 | Length |
| byte 10 | 05 | Operation: ACTION |
| byte 11 | 30 | SEQUENCE (Parameters) |
| byte 12 | 0C | Length |
| byte 13 | 30 | SEQUENCE (ResourceID) |
| byte 14 | 07 | Length |
| byte 15 | 06 | OBJECT ID: Universal, primitive tag= 16 |
| byte 16 | 05 | Length |
| byte 17 | 2B | {ISO (1), identified organization (3)} |
| byte 18 | XX | {ICD} |
| byte 19 | XX | {Organization Code} |
| byte 20 | 01 | Channel: B1 |
| byte 21 | 00 | Circuit mode |
| byte 22 | 86 | [6] IMPLICIT INTEGER (Action Type) |
| byte 23 | 01 | Length |
| byte 24 | 08 | Test Call |

## E.2 Example for SET (T200 to 1,5 seconds, N200 to 4 and N201 to 192)

| | | |
|---|---|---|
| byte 1 | A1 | Class: Context - Constructor - Invoke |
| byte 2 | 21 | Length: 33 |
| byte 3 | 30 | SEQUENCE: Universal, Constructor tag = 16. |
| byte 4 | 1F | Length: 31 |
| byte 5 | 02 | INTEGER |
| byte 6 | 01 | Length: 1 |
| byte 7 | InvokeID | range 0 - 255 |
| byte 8 | 02 | INTEGER |
| byte 9 | 01 | Length |
| byte 10 | 04 | Operation: SET |
| byte 11 | 30 | SEQUENCE (Parameter) |
| byte 12 | 17 | Length: 23 |
| byte 13 | 30 | SEQUENCE (ResourceID) |
| byte 14 | 08 | Length |
| byte 15 | 06 | OBJECT ID |
| byte 16 | 06 | Length |
| byte 17 | 2B | {ISO (1), identified organization (3)} |
| byte 18 | XX | {ICD} |
| byte 19 | XX | {Organization Code} |
| byte 20 | FF | Channel: D |
| byte 21 | 00 | Signalling |
| byte 22 | 02 | Layer 2 |
| byte 23 | 31 | [UNIVERSAL 17] SET OF (Attribute List) |
| byte 24 | 0B | Length: 11 |
| byte 25 | A7 | [CONTEXT 7] IMPLICIT SET |
| byte 26 | 09 | Length: 9 |
| byte 27 | 81 | [CONTEXT 1] IMPLICIT INTEGER: T200 |
| byte 28 | 01 | Length |
| byte 29 | 0F | T200 Value: 15 (in tenth of seconds) |
| byte 30 | 82 | [CONTEXT 2] IMPLICIT INTEGER: N200 |
| byte 31 | 01 | Length |
| byte 32 | 04 | N200 Value: 4 |
| byte 33 | 83 | [CONTEXT 3] IMPLICIT INTEGER: N201 |
| byte 34 | 01 | Length |
| byte 35 | C0 | N201 Value: 192 |

## E.3 Example for ACTION (Loopback Activation at TE - Loopback Reference B1)

| | | |
|---|---|---|
| byte 1 | A1 | Class: Context - Constructed - Invoke |
| byte 2 | 1E | Length: 30 bytes |
| byte 3 | 30 | SEQUENCE: Universal, Constructor tag = 16. |
| byte 4 | 1C | Length: 28 |
| byte 5 | 02 | INTEGER |
| byte 6 | 01 | Length: 1 |
| byte 7 | InvokeID | range 0 - 255 |
| byte 8 | 02 | INTEGER |
| byte 9 | 01 | Length |
| byte 10 | 05 | Operation: ACTION |
| byte 11 | 30 | SEQUENCE (Parameters) |
| byte 12 | 14 | Length: 20 |
| byte 13 | 30 | SEQUENCE (ResourceID) |
| byte 14 | 05 | Length |
| byte 15 | 06 | OBJECT ID |
| byte 16 | 03 | Length |
| byte 17 | 2B | {ISO (1), identified organization (3)} |
| byte 18 | XX | {ICD} |
| byte 19 | XX | {Organization Code} |
| byte 20 | 86 | IMPLICIT INTEGER (Action Type) |
| byte 21 | 01 | Length |
| byte 22 | 01 | Loop activation |
| byte 23 | 31 | SET OF |
| byte 24 | 08 | Length |
| byte 25 | A7 | [7] IMPLICIT SET |
| byte 26 | 06 | Length |
| byte 27 | 82 | Loop Reference |
| byte 28 | 01 | Length |
| byte 29 | 05 | Reference: B1 |
| byte 30 | 83 | Loop Timer |
| byte 31 | 01 | Length |
| byte 32 | F0 | 20 Minutes (Timer unit: 5 seconds) |

## E.4 Example for EVENT REPORT (Channel B3, Layer 1, Circuit Mode, Loss of Synchronization)

| | | |
|---|---|---|
| byte 1 | A1 | Class: Context - Constructed - Invoke |
| byte 2 | 1E | Length: 30 bytes |
| byte 3 | 30 | SEQUENCE: Universal, Constructor tag = 16. |
| byte 4 | 1C | Length: 28 |
| byte 5 | 02 | INTEGER |
| byte 6 | 01 | Length: 1 |
| byte 7 | InvokeID | range 0 - 255 |
| byte 8 | 02 | INTEGER |
| byte 9 | 01 | Length |
| byte 10 | 02 | Operation: EVENT REPORT |
| byte 11 | 30 | SEQUENCE (Parameters) |
| byte 12 | 14 | Length: 20 |
| byte 13 | 30 | SEQUENCE (ResourceID) |
| byte 14 | 08 | Length |
| byte 15 | 06 | OBJECT ID |
| byte 16 | 06 | Length |
| byte 17 | 2B | {ISO (1), identified organization (3)} |
| byte 18 | XX | {ICD} |
| byte 19 | XX | {Organization Code} |
| byte 20 | 03 | Channle ID (B3) |
| byte 21 | 00 | Circuit Mode |
| byte 22 | 01 | Layer 1 |
| byte 23 | 81 | Defined Event: IMPLICIT INTEGER |
| byte 24 | 01 | Length |
| byte 25 | 02 | Loss of Synch |
| byte 26 | 28 | EXTERNAL: [Universal 8] IMPLICIT SEQUENCE |
| byte 27 | 05 | Length |
| byte 28 | 81 | [1] IMPLICIT OCTET STRING |
| byte 29 | 03 | Length |
| byte 30 | 10 | 25 December 23:58:25 |
| byte 31 | A5 | (31017504 seconds after the reference time) |
| byte 32 | EC | |

As an example the time coding above is indicated in units of two seconds starting from the beginning of the year, i.e. from the 1st January 1988 at 00:00:00 o'clock.

## APPENDIX F

## LIST OF ACRONYMS

| | |
|---|---|
| A- | Association- |
| ACSE | Application Control Service Elements |
| APDU | Application Protocol Data Unit |
| ASN.1 | Abstract Syntax Notation ONE |
| BC | Bearer Capability |
| CEI | Connection Endpoint Identifier |
| CES | Connection Endpoint Suffix |
| CESP | Compatible Equipment Selection Parameter |
| CL | Connectionless |
| CMIP | Common Management Information Protocol |
| CMIS | Common Management Information Service |
| CNF | Confirmation |
| CO | Connection-Oriented |
| CR | Call Reference |
| DPE | Data Processing Equipment |
| EoC | End of Contents |
| IND | Indication |
| LLC | Lower Layer Compatibility |
| LME | Layer Management Entity |
| LMI | Layer Management Interface |
| M- | Management- |
| MIP | Management Information Protocol |
| N- | Network- |
| OSI | Open Systems Interconnection |
| OSMAE | Open System Management Application Entity |
| OSMAP | Open System Management Application Process |
| P- | Presentation- |
| PCI | Protocol Control Information |
| PDU | Protocol Data Unit |
| PE | Protocol Entity |
| PSN | Private Switching Network |
| PT | PSN Termination |
| QoS | Quality of Service |
| RO | Remote Operation |
| ROS | Remote Operation Service |
| ROSE | Remote Operation Service Elements |
| RQ | Request |
| RSP | Response |

| | |
|---|---|
| SAP | Service Access Point |
| SAPI | Service Access Point Identifier |
| SEI | Service Endpoint Identifier |
| SM | System Management |
| SMDSI | System Management Data Service Interface |
| SMI | System Management Interface |
| TE | Terminal Equipment |
| UEI | Unique Equipment Identifier |
| UUI | User-to-User Information |