INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.812

(02/2004)

SERIES Q: SWITCHING AND SIGNALLING

Q3 interface

Upper layer protocol profiles for the Q and X interfaces

ITU-T Recommendation Q.812

ITU-T Recommendation Q.812

Upper layer protocol profiles for the Q and X interfaces

Summary

This Recommendation provides the upper layer (5-7) protocol profiles for the Q and X interfaces as defined in the M.3000 series of ITU-T Recommendations.

Source

ITU-T Recommendation Q.812 was approved on 13 February 2004 by ITU-T Study Group 4 (2001-2004) under the ITU-T Recommendation A.8 procedure.

CONTENTS

Page

ITU-T Recommendation Q.812

Upper layer protocol profiles for the Q and X interfaces

1 Scope

This Recommendation defines the characteristics of protocol profiles for the Q and X interfaces, as defined in the M.3000 series of ITU-T Recommendations. The interface will support bidirectional data transfer for the management of telecommunication systems.

The need for security functionality is recognized, but is not fully addressed in this Recommendation and is for further study. Users may need to use mechanisms outside this Recommendation in order to address their specific security needs. Security mechanisms chosen may depend on the network configuration being used.

This Recommendation defines:

– the layer services profiles;
– the layer protocols profiles;
– the application service and protocols profiles;
– the conformance requirements to be met by an implementation of this interface.

This Recommendation does not define:

– the structure or meaning of the management information that is transmitted by means of the protocol suite;
– the manner in which management is accomplished as a result of the application protocol exchanges;
– the interactions which result in the use of the application layer protocols.

The profiles in this Recommendation align with equivalent ISPs.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[1] ISO/IEC/TR 10000-1:1995, *Information technology – Framework and taxonomy of International Standardized Profiles – Part 1: General principles and documentation framework*.

[2] ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The basic model*.

[3] ITU-T Recommendation M.3010 (2000), *Principles for a telecommunications management network*.

[4] ITU-T Recommendation X.224 (1995) | ISO/IEC 8073:1997, *Information technology – Open Systems Interconnection – Protocol for providing the connection-mode transport service*.

[5] ITU-T Recommendation X.225 (1995) | ISO/IEC 8327-1:1996, *Information technology – Open Systems Interconnection – Connection-oriented Session protocol: Protocol specification, plus* Amendment 1 (1997): *Efficiency enhancements.*

[6] ISO/IEC ISP 11183-1:1992, *Information technology – International Standardized Profiles AOM1n OSI Management – Management Communications – Part 1: Specification of ACSE, presentation and session protocols for the use by ROSE and CMISE.*

[7] ITU-T Recommendation X.216 (1994) | ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition.*

[8] ITU-T Recommendation X.226 (1994) | ISO/IEC 8823-1:1994, *Information technology – Open Systems Interconnection – Connection-oriented Presentation protocol: Protocol specification.*

[9] ITU-T Recommendation X.209 (1988), *Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1).*

ISO/IEC 8825:1990, *Information technology – Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

[10] ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

[11] ITU-T Recommendation X.681 (1994) | ISO/IEC 8824-2:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*

[12] ITU-T Recommendation X.682 (1994) | ISO/IEC 8824-3:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*

[13] ITU-T Recommendation X.683 (1994) | ISO/IEC 8824-4:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Parametrization of ASN.1 specifications.*

[14] ITU-T Recommendation X.208 (1988), *Specification of Abstract Syntax Notation One (ASN.1).*

ISO/IEC 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1).*

[15] ISO/IEC 9545:1994, *Information technology – Open Systems Interconnection – Application Layer structure.*

[16] ITU-T Recommendation X.217 (1995) | ISO/IEC 8649:1996, *Information technology – Open Systems Interconnection – Service definition for the association control service element.*

[17] ITU-T Recommendation X.227 (1995) | ISO/IEC 8650-1:1996, *Information technology – Open Systems Interconnection – Connection-oriented protocol for the association control service element: Protocol specification.*

[18] ITU-T Recommendation X.219 (1988), *Remote operations: Model, notation and service definition.*

ISO/IEC 9072-1:1989, *Information processing systems – Text communication – Remote operations – Part 1: Model, notation and service definition.*

[19] ITU-T Recommendation X.229 (1988), *Remote operations: Protocol specification.*

ISO/IEC 9072-2:1989, *Information processing systems – Text communication – Remote operations – Part 2: Protocol specification.*

[20] ITU-T Recommendation X.710 (1991), *Common management information service definition for CCITT applications.*

ISO/IEC 9595:1991, *Information technology – Open Systems Interconnection – Common management information service definition.*

[21] ITU-T Recommendation X.711 (1991), *Common management information protocol specification for CCITT applications.*

ISO/IEC 9596-1:1991, *Information technology – Open Systems Interconnection – Common management information protocol – Part 1: Specification.*

[22] ISO/IEC ISP 11183-3:1992, *Information technology – International Standardized Profiles AOM1n OSI Management – Management Communications – Part 3: CMISE/ROSE for AOM11 – Basic Management Communications.*

[23] ISO/IEC ISP 11183-2:1992, *Information technology – International Standardized Profiles AOM1n OSI Management – Management Communications – Part 2: CMISE/ROSE for AOM12 – Enhanced Management Communications.*

[24] ISO/IEC ISP 10607-1:1995, *Information technology – International Standardized Profiles AFTnn – File Transfer, Access and Management – Part 1: Specification of ACSE, Presentation and Session protocols for the use of FTAM.*

[25] ISO 8571-1:1988, *Information processing systems – Open Systems Interconnection – File Transfer, Access and Management – Part 1: General introduction.*

[26] ISO 8571-2:1988, *Information processing systems – Open Systems Interconnection – File Transfer, Access and Management – Part 2: Virtual Filestore Definition.*

[27] ISO 8571-3:1988, *Information processing systems – Open Systems Interconnection – File Transfer, Access and Management – Part 3: File Service Definition.*

[28] ISO 8571-4:1988, *Information processing systems – Open Systems Interconnection – File Transfer, Access and Management – Part 4: File Protocol Specification.*

[29] ITU-T Recommendation X.500 (1997) | ISO/IEC 9594-1:1998, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services.*

[30] ITU-T Recommendation X.501 (1997) | ISO/IEC 9594-2:1998, *Information technology – Open Systems Interconnection – The Directory: Models.*

[31] ITU-T Recommendation X.511 (1997) | ISO/IEC 9594-3:1998, *Information technology – Open Systems Interconnection – The Directory: Abstract service definition.*

[32] ITU-T Recommendation X.518 (1997) | ISO/IEC 9594-4:1998, *Information technology – Open Systems Interconnection – The Directory: Procedures for distributed operation.*

[33] ITU-T Recommendation X.519 (1997) | ISO/IEC 9594-5:1998, *Information Technology – Open Systems Interconnection – The Directory: Protocol specifications.*

[34] ITU-T Recommendation X.520 (1997) | ISO/IEC 9594-6:1998, *Information technology – Open Systems Interconnection – The Directory: Selected attribute types.*

[35] ITU-T Recommendation X.521 (1997) | ISO/IEC 9594-7:1998, *Information technology – Open Systems Interconnection – The Directory: Selected object classes.*

[36] ITU-T Recommendation X.509 (1997) | ISO/IEC 9594-8:1998, *Information technology – Open Systems Interconnection – The Directory: Authentication framework.*

[37] ITU-T Recommendation X.880 (1994) | ISO/IEC 13712-1:1995, *Information technology – Remote Operations: Concepts, model and notation.*

[38] ITU-T Recommendation X.881 (1994) | ISO/IEC 13712-2:1995, *Information technology – Remote Operations: OSI realizations – Remote Operations Service Element (ROSE) service definition.*

[39] ITU-T Recommendation X.830 (1995) | ISO/IEC 11586-1:1996, *Information technology – Open Systems Interconnection – Generic upper layers security: Overview, models and notation*.

[40] ISO/IEC ISP 10607-3:1995, *Information technology – International Standardized Profiles AFTnn – File Transfer, Access and Management – Part 3: AFT11 – Simple File Transfer Service (unstructured)*.

[41] ITU-T Recommendation X.214 (1995) | ISO/IEC 8072:1996, *Information technology – Open Systems Interconnection – Transport service definition*.

[42] ITU-T Recommendation X.882 (1994) | ISO/IEC 13712-3:1995, *Information technology – Remote Operations: OSI realizations – Remote Operations Service Element (ROSE) protocol specification*.

[43] ITU-T Recommendation X.800 (1991), *Security architecture for Open Systems Interconnection for CCITT applications*.

ISO/IEC 7498-2:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture*.

[44] ITU-T Recommendation X.803 (1994) | ISO/IEC 10745:1995, *Information technology – Open Systems Interconnection – Upper layers security model*.

[45] ITU-T Recommendation Q.811 (2004), *Lower layer protocol profiles for the Q and X interfaces*.

[46] ITU-T Recommendation Q.814 (2000), *Specification of an electronic data interchange interactive agent*.

[47] ITU-T Recommendation Q.815 (2000), *Specification of a security model for whole message protection*.

[48] IETF RFC 3410 (2002), *Introduction and Applicability Statements for Internet Standard Management Framework*.

[49] IETF RFC 3411 (2002), *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*.

[50] IETF RFC 3412 (2002), *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)*.

[51] IETF RFC 3413 (2002), *Simple Network Management Protocol (SNMP) Applications*.

[52] IETF RFC 3414 (2002), *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*.

[53] IETF RFC 3415 (2002), *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)*.

[54] IETF RFC 3416 (2002), *Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)*.

[55] IETF RFC 3417 (2002), *Transport Mappings for the Simple Network Management Protocol (SNMP)*.

[56] IETF RFC 3584 (2003), *Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework*.

[57] IETF RFC 2578 (1999), *Structure of Management Information Version 2 (SMIv2)*.

[58] IETF RFC 3430 (2002), *Simple Network Management Protocol (SNMP) over Transmission Control Protocol (TCP) Transport Mapping*.

[59] IETF RFC 2401 (1998), *Security Architecture for the Internet Protocol.*

[60] ISO/IEC ISP 10608-1:1992, *Information technology – International Standardized Profile TAnnnn – Connection-mode Transport Service over Connectionless-mode Network Service – Part 1: General overview and subnetwork-independent requirements*.

[61] ISO/IEC ISP 10609-1:1992, *Information technology – International Standardized Profiles TB, TC, TD and TE – Connection-mode Transport Service over connection-mode Network Service – Part 1: Subnetwork-type independent requirements for Group TB*.

[62] CORBA GIOP Specification, Chapter 15 of *The Common Object Request Broker: Architecture and Specification*, Revision 2.3, Object Management Group (OMG Doc. Number: Formal/98-12-01).

[63] CORBA Security Service Specification, Chapter 15 of *CORBA services: Common Object Services Specification*, Object Management Group (OMG Doc. Number: Formal/98-12-17).

[64] ITU-T Recommendation M.3030 (2002), *Telecommunications Markup Language (tML) framework.*

3 Definitions

This Recommendation defines the following terms:

3.1 International Standardized Profile (ISP): An internationally agreed-to, harmonized document which identifies a standard or group of standards, together with options and parameters, necessary to accomplish a function or a set of functions [1].

3.2 Interactive Agent (IA) (ITU-T Rec. Q.814): The IA supports the exchange of electronic data interchange (UN/EDIFACT or ASC X12 EDI) transactions between peer entities. The IA functions as an interface between its direct user (normally an EDIFACT/ASC X12 EDI translator or a security module) and the transport layer security. Various implementation approaches may be taken ranging from a simple API (Application Program Interface) through a stand-alone program. The IA is described in ITU-T Rec. Q.814 and the Security Module is described in ITU-T Rec. 815.

3.3 transport layer security (ITU-T Rec. Q.814): The Transport Layer Security (TLS) protocol optionally provides communications privacy. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and intrusion. The TLS protocol also provides strong peer authentication and data flow integrity.

3.4 Distinguished Encoding Rules (DER) (ITU-T Rec. X.690): DERs for ASN.1 are a subset of Basic Encoding Rules (BER), and give exactly one way to represent any ASN.1 value as an octet string. DERs are intended for applications in which a unique octet string encoding is needed, as is the case when an ASN.1 message is encoded for transport via TLS. DERs are described in ITU-T Rec. X.690. In this profile the definite-length method of constructing IA messages must be employed.

3.5 Interactive Agent Transfer Protocol (IATP) (ITU-T Rec. Q.814): The IATP protocol is utilized between peer Interactive Agents wishing to exchange Electronic Data Interchange transactions/messages via Transmission Control Protocol/Internet Protocol utilizing Transport Layer Security. The IATP is described in ITU-T Rec. Q.814.

3.6 Electronic Data Interchange (EDI) translator (ITU-T Rec. Q.814): A computer software module or program that translates private data formats and representations to/from standard formats and standard data representations such as those specified by ISO 9735 or ANSI ASC X12.

4 Abbreviations

This Recommendation uses the following abbreviations.

ACSE	Association Control Service Element
AE	Application Entity
APDU	Application Protocol Data Unit
ASE	Application Service Element
ASN.1	Abstract Syntax Notation One
ASO	Application Service Object
BER	Basic Encoding Rules
CF	Control Function
CLNS	ConnectionLess-mode Network layer Service
CMIP	Common Management Information Protocol
CMISE	Common Management Information Service Element
CONS	Connection-mode Network layer Service
CORBA	Common Object Request Broker Architecture
COTS	Connection-mode Transport Service
DAP	Directory Access Protocol
DCN	Data Communication Network
DER	Distinguished Encoding Rules
DSA	Directory System Agent
DUA	Directory User Agent
EDI	Electronic Data Interchange
EDIFACT	Electronic Data Interchange For Administration, Commerce and Transport
FTAM	File Transfer, Access and Management
GULS	Generic Upper Layer Security
IA	Interactive Agent
IATP	Interactive Agent Transfer Protocol
IDL	Interface Definition Language
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPSec	Security Architecture for the IP Protocol
ISO	International Organization for Standardization
ISP	International Standardized Profile
ITU	International Telecommunication Union
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
NBS	National Bureau of Standards

NCMS	Network Connection Management Subprotocol
NE	Network Element
OS	Operations System
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
RFC	Request for Comments
ROS	Remote Operations Service
ROSE	Remote Operations Service Element
SACF	Single Association Control Function
SMASE	Systems Management Application Service Element
SNMP	Simple Network Management Protocol
SPDU	Session Protocol Data Unit
TCP	Transmission Control Protocol
TLS	Transport Layer Security
tML	Telecommunications Markup Language
TMN	Telecommunications Management Network
UDP	User Datagram Protocol
USM	User-based Security Model

5 Upper layer protocol specifications for the OSI paradigm

5.1 Introduction to Upper layer protocol specifications for the OSI paradigm

The communication services and protocols referred to in this Recommendation are in accordance with the Open Systems Interconnection (OSI) Reference Model [2].

The protocols for the different layers are based on ITU-T Recommendations and/or ISO Standards, CORBA Specifications from the OMG and Internet Protocol Specifications from the IETF.

Three types of protocol profiles are defined in this Recommendation:

– upper layer protocol profile for Interactive Class services;
– upper layer protocol profile for File-oriented Class services;
– upper layer protocol specification for Directory services.

The three protocol profiles can be applied to applications using DCN, as defined by ITU-T Rec. M.3010 [3].

Interface Q is defined to intend to connect Mediation Devices to Operations Systems (OSs), Q Adapters to OSs, NEs to OSs, and OSs to OSs via a DCN. The X Interface is defined to connect the TMNs of two Administrations.

Other ASEs will be added in the identified protocol profiles as new requirements develop.

5.2 Upper layer protocol specification for Interactive Class services

Figure 1 illustrates the protocol stack of the upper layer protocol profile for Interactive Class services. The profile for TMN function sets corresponding to the SMASE for Interactive Class of services may be specified as part of Recommendations defining the information models and services.

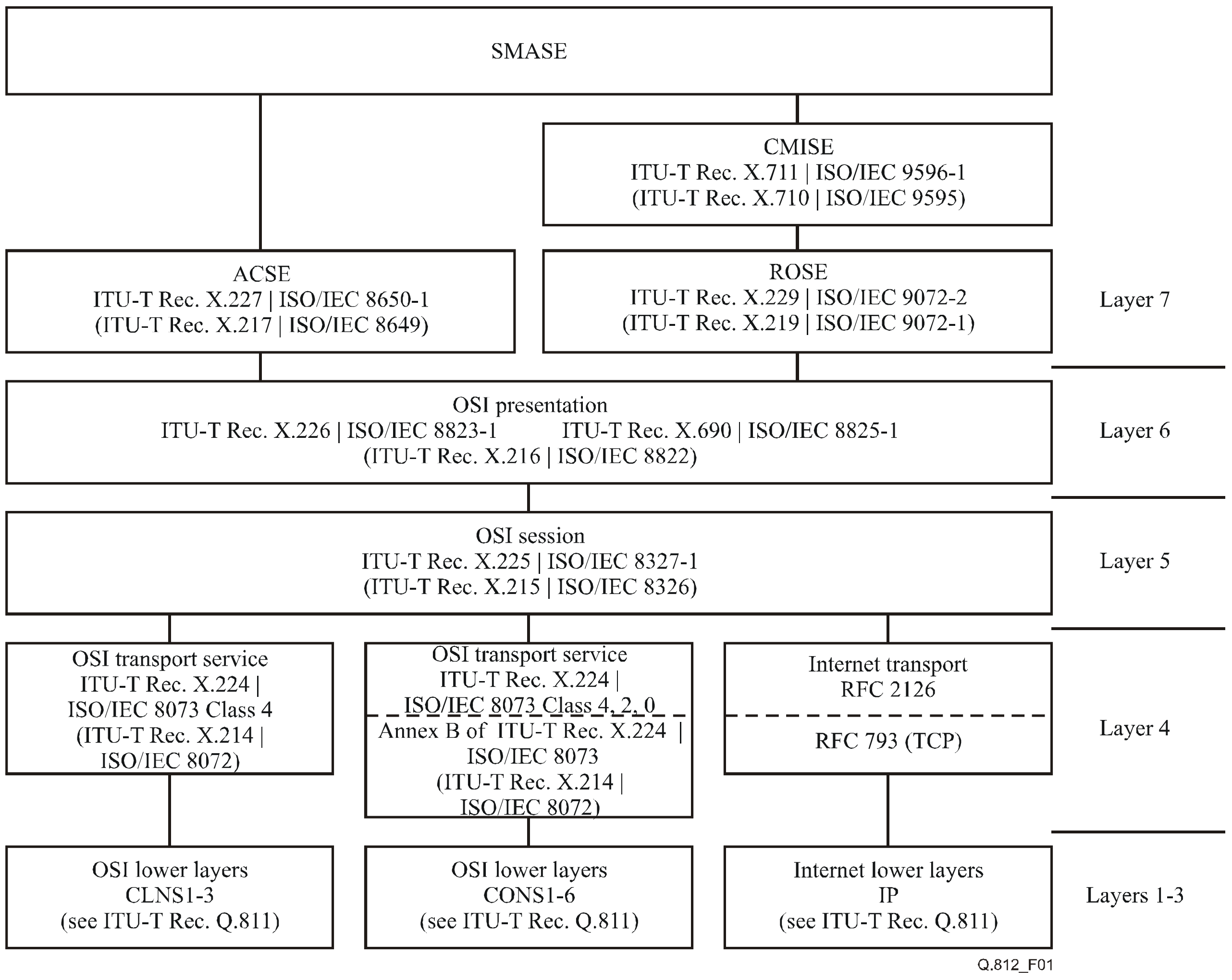
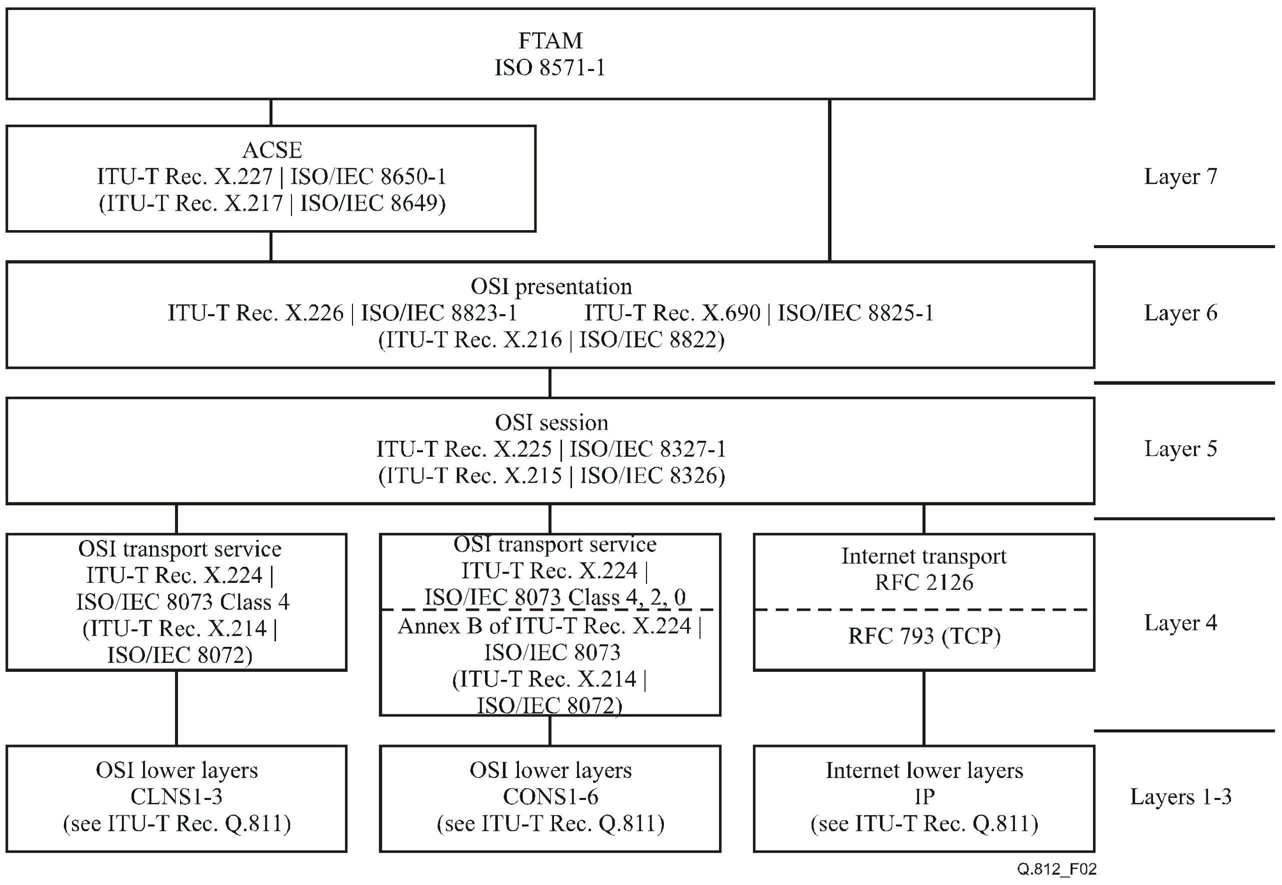


Figure 1/Q.812 – Protocol stack of the upper layer protocol profile for Interactive Class services for the OSI paradigm

5.3 Upper layer protocol specification for File-oriented Class services

Figure 2 illustrates the protocol stack of the upper layer protocol profile for File-oriented Class services.

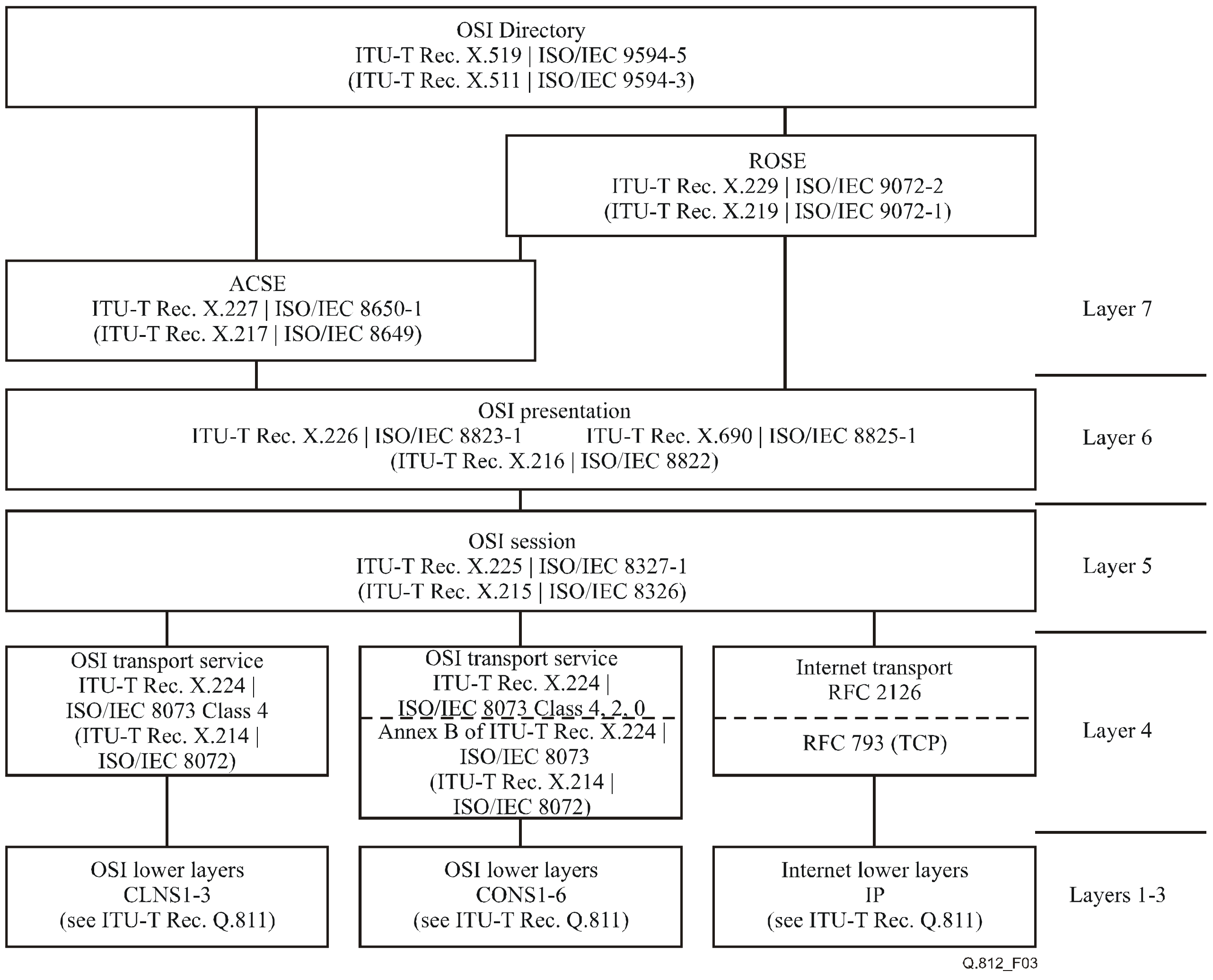


FTAM
ISO 8571-1
ACSE
ITU-T Rec. X.227 | ISO/IEC 8650-1
(ITU-T Rec. X.217 | ISO/IEC 8649)
Layer 7
OSI presentation
ITU-T Rec. X.226 | ISO/IEC 8823-1
ITU-T Rec. X.690 | ISO/IEC 8825-1
(ITU-T Rec. X.216 | ISO/IEC 8822)
Layer 6
OSI session
ITU-T Rec. X.225 | ISO/IEC 8327-1
(ITU-T Rec. X.215 | ISO/IEC 8326)
Layer 5
OSI transport service
ITU-T Rec. X.224 | ISO/IEC 8073 Class 4
(ITU-T Rec. X.214 | ISO/IEC 8072)
OSI transport service
ITU-T Rec. X.224 | ISO/IEC 8073 Class 4, 2, 0
Annex B of ITU-T Rec. X.224 | ISO/IEC 8073
(ITU-T Rec. X.214 | ISO/IEC 8072)
Internet transport
RFC 2126
RFC 793 (TCP)
Layer 4
OSI lower layers
CLNS1-3
(see ITU-T Rec. Q.811)
OSI lower layers
CONS1-6
(see ITU-T Rec. Q.811)
Internet lower layers
IP
(see ITU-T Rec. Q.811)
Layers 1-3
Q.812_F02

Figure 2/Q.812 – Protocol stack of the upper layer protocol profile for File-oriented Class services for the OSI paradigm

5.4 Upper layer protocol specification for Directory services

Figure 3 illustrates the protocol stack of the upper layer protocol profile for Directory services.



OSI Directory
ITU-T Rec. X.519 | ISO/IEC 9594-5
(ITU-T Rec. X.511 | ISO/IEC 9594-3)
ROSE
ITU-T Rec. X.229 | ISO/IEC 9072-2
(ITU-T Rec. X.219 | ISO/IEC 9072-1)
ACSE
ITU-T Rec. X.227 | ISO/IEC 8650-1
(ITU-T Rec. X.217 | ISO/IEC 8649)
Layer 7
OSI presentation
ITU-T Rec. X.226 | ISO/IEC 8823-1
ITU-T Rec. X.690 | ISO/IEC 8825-1
(ITU-T Rec. X.216 | ISO/IEC 8822)
Layer 6
OSI session
ITU-T Rec. X.225 | ISO/IEC 8327-1
(ITU-T Rec. X.215 | ISO/IEC 8326)
Layer 5
OSI transport service
ITU-T Rec. X.224 | ISO/IEC 8073 Class 4
(ITU-T Rec. X.214 | ISO/IEC 8072)
OSI transport service
ITU-T Rec. X.224 | ISO/IEC 8073 Class 4, 2, 0
Annex B of ITU-T Rec. X.224 | ISO/IEC 8073
(ITU-T Rec. X.214 | ISO/IEC 8072)
Internet transport
RFC 2126
RFC 793 (TCP)
Layer 4
OSI lower layers
CLNS1-3
(see ITU-T Rec. Q.811)
OSI lower layers
CONS1-6
(see ITU-T Rec. Q.811)
Internet lower layers
IP
(see ITU-T Rec. Q.811)
Layers 1-3
Q.812_F03

Figure 3/Q.812 – Protocol stack of the upper layer protocol profile for Directory services in the OSI paradigm

5.5 Upper layer protocol specification for store and forward services

The upper layer protocols to be used for store and forward services (e.g., for exchange of EDI formatted information) is for further study.

6 Upper layer protocol specification for Interactive Class services using the OSI paradigm

6.1 Transport layer profiles

6.1.1 Transport layer profile for CLNS1, CLNS2 and CLNS3

This clause defines the transport layer profile for use with CLNS1, CLNS2 and CLNS3 as defined in ITU-T Rec. Q.811 [45].

6.1.1.1 Services profile

It is mandatory that for the connectionless-mode Network service, the Transport service shall conform to ITU-T Rec. X.214 | ISO/IEC 8072 [41].

6.1.1.2 Protocol profile

Operation of the Transport protocol over the connectionless-mode network layer service (CLNS), as described in ITU-T Rec. X.213 | ISO/IEC 8348, shall use the elements of ITU-T Rec. X.224 | ISO/IEC 8073 [4], Class 4 operation over the CLNS. Provisions in 6.1.2.1.2, 6.1.2.1.3, 6.1.2.1.4, 6.1.2.2.4, 6.1.2.2.6, 6.1.2.2.7 and 6.1.2.2.8, of CONS1 profile Transport protocol also apply here.

6.1.1.3 Class of service

Support of Class 4 operation of ITU-T Rec. X.224 | ISO/IEC 8073 [4] is mandatory.

6.1.1.4 Transport layer attributes

Transport layer attributes for Class 4 operation over the Connectionless-mode Network layer Service shall be as shown in Table 1.

Table 1/Q.812 – Transport layer attributes [for use with Connectionless-mode Network layer Service (CLNS)]

	Value/Range/Option	**Default**
Maximum TPDU (Octets)	128, 256, 512, 1024 (2048, 4096, 8192 optional)	(128)
TSAP-ID (Note 1)	Up to 32 octets	
Class of service	4	–
Preferred class	4	–
Alternative Class	None	– –
Expedited Data	Non-use	–
Options:		
Security Parameters	Optional	–
Data TPDU numbering (Note 2)	Normal, extended	(Normal)
Checksum (Note 3)	Use, non-use	(Non-use)
Parameters:		
T1 – Retransmission time	0.25-64 seconds (Note 4)	(8)
N – Retransmissions	2-15	(2)
L – Bound on reference	1-256 seconds	(32)
I – Inactivity time	2-512 seconds	(64)

NOTE 1 – Some systems may require TSAP-IDs. However, all systems shall be capable of generating called TSAP-IDs in CR TPDUs and capable of receiving calling and called TSAP-IDs in received CR and CC TPDUs, respectively.

NOTE 2 – Extended format option shall be implemented. Non-use of this option shall be negotiable. The responder shall honour the initiator's request whenever possible. Negotiation to other than what has been requested shall only occur under abnormal conditions, for example, severe congestion, as determined by the implementor. Initiators shall be prepared to operate in the mode confirmed by the responder.

NOTE 3 – Use of checksum is required for the CR TPDU. An additional requirement is that all implementations shall support the negotiated "non-use" of the checksum. Initiators shall request and responders shall agree to "non-use" of the checksum.

NOTE 4 – The Transport layer T1 timer value should always be greater than the link layer T1 timer value.

6.1.2 Transport layer profile for CONS1, CONS2, CONS3 and CONS5

This clause defines the transport layer profile for use with CONS1, CONS2, CONS3, CONS4 and CONS5 as defined in ITU-T Rec. Q.811 [45].

6.1.2.1 Services profiles

The Protocol Profiles described in this Recommendation provide the Connection-mode Transport Service (COTS) to the OSI upper layers that is defined in ITU-T Rec. X.214 | ISO/IEC 8072 [41].

6.1.2.1.1 Splitting

Responders may refuse network connections which could impose an unnecessary restriction on the ability to establish outgoing network connections. To prevent repeated ineffective attempts during splitting, initiators shall refrain from immediately requesting additional network connections for a transport connection after a network connection has been refused. The time delay before requesting additional network connections is for further study.

6.1.2.1.2 Quality of service negotiation

Quality of service negotiation is outside the scope of this Recommendation. If quality of service negotiation is not supported, receipt of the parameters "throughput", "residual error rate", "priority", and "transit delay" in the CR and CC TPDUs shall be ignored.

6.1.2.1.3 TPDU size negotiation

Interoperability is achieved by having the initiator propose a TPDU size from the set specified in Table 2 and by the responder selecting the most appropriate TPDU size between 128 and the proposed TPDU size. The rules for negotiation of the size of the TPDU to be used in a given instance of communication are specified in ITU-T Rec. X.224 | ISO/IEC 8073 [4].

The choice of TPDU size is a local implementation issue.

6.1.2.1.4 Negotiation of protection

Negotiation of protection is outside the scope of this Recommendation. If negotiation of protection is not supported, receipt of the protection parameters in any CR TPDU and any CC TPDU shall be ignored.

6.1.2.2 Protocol profile

It is mandatory that, for the connection-mode network service, the Transport protocol shall conform to ITU-T Rec. X.224 | ISO/IEC 8073 [4] and to those provisions, therefore that apply to the use of the Connection-mode Network layer Service (CONS).

6.1.2.2.1 Class of service

Classes 4, 2, and 0 shall be supported as shown in Table 2 in countries requiring the features of Transport layer Class 4. The conformance rules of ITU-T Rec. X.224 | ISO/IEC 8073 [4] require that Classes 0 and 2 be supported as well when Class 4 is specified. For existing equipment and in countries not requiring Class 4, support of Class 0 is mandatory and Class 2 is optional.

The default values shall be part of a vendor's offering. That is, unless otherwise specified by the user, the default parameters shall be the initial values supplied. They can be subsequently changed by the user within the specified range.

In addition to the requirements specified in ITU-T Rec. X.224 | ISO/IEC 8073 [4], equipment shall meet the following requirement: if a responder receives an alternate class of "none", it shall respond with the preferred class. Rules for responders are specified in Table 3.

User options shall be provided to designate the preferred and alternate classes (see Table 3/X.224 | ISO/IEC 8073 [4]). When all of the classes are supported, the preferred class for connection is Class 4.

Table 2/Q.812 – Transport layer attributes
[for Connection-mode Network layer Service (CONS)]

Attribute	Range	Default
Maximum TPDU (octets)	128, 256, 512, 1024 (2048, 4096, 8192 optional)	(128)
Class of Service Preferred Class Alternative Class Expedited Data	4, 2, 0 4, 2, 0 4, 2, 0, none Non-use	 (4) (None)
Options for Class 4 Data TPDU numbering (Note 2)	 Normal, extended	 (Normal)
Options for Class 2 Data TPDU numbering (Note 2)	 Normal, extended	 (Normal)
Flow control	Explicit	
Parameters for Class 4 T1 – Retransmission time	 0.25-64 seconds (Note 4)	 (8)
N – Retransmissions	2 (other values for further study)	
L – Bound on reference	1-256 seconds	(32)
I – Inactivity time	2-512 seconds	(64)
NOTE 1 – Some systems may require TSAP-IDs. However, all systems shall be capable of generating called TSAP-IDs in CR TPDUs and capable of receiving calling and called TSAP-IDs in received CR and CC TPDUs, respectively. NOTE 2 – Extended format option shall be implemented. Non-use of this option shall be negotiable. The responder shall honour the initiator's request whenever possible. Negotiation to other than what has been requested shall only occur under abnormal conditions, for example, severe congestion, as determined by the implementor. Initiators shall be prepared to operate in the mode confirmed by the responder. NOTE 3 – Use of the checksum is required for the CR TPDU. An additional requirement is that all implementations shall support the negotiated "non-use" of the checksum. Initiators shall request and responders shall agree to "non-use" of the checksum. NOTE 4 – The Transport layer T1 timer should always be greater than the link layer T1 timer.		

Table 3/Q.812 – Valid response corresponding to preferred and any alternative class proposed in the CR TPDU

Preferred class	Alternative class			
	0	2	4	None
0	Not valid	Not valid	Not valid	Class 0
2	Classes 0, 2	Class 2	Not valid	Class 2
4	Classes 0, 2, 4	Class 2 or 4	Class 4	Class 2 or 4

6.1.2.2.2 Protocol identification

For the purpose of Transport layer protocol identification, the procedures specified in Annex B of ITU-T Rec. X.224 | ISO/IEC 8073 [4] and ITU-T Rec. X.264 | ISO/IEC 11570 shall be used. The conventions for protocol identification given in ITU-T Rec. X.263 |ISO/IEC TR 9577 should be followed. Selection of codes not specified in the referenced standards is for further study. The absence of call user data in a call request or call accept packet of ITU-T Rec. X.25 and ISO/IEC 8208 indicates the operation of the Transport layer procedures of ITU-T Rec. X.224 | ISO/IEC 8073 [4].

6.1.2.2.3 Attributes

Attributes of the Transport layer for use with CONS are summarized in Table 2. The selection of values within required and optional ranges depends on characteristics of the messages.

NOTE – The need to support high priority messages that require low transit delay on a given Transport connection must be reflected in the Quality of Service parameters requested when the Transport connection is established. A properly implemented Transport entity should not multiplex high priority messages that require low transit delay if it cannot provide the requested Quality of Service. Since this is an implementation detail, it is not subject to standardization.

6.1.2.2.4 User data in connection request and connection confirm TPDUs

User data in the connection request and connection confirm TPDUs are optional in ITU-T Rec. X.224 | ISO/IEC 8073 [4]. No Transport service user shall send it: all protocol implementations shall be prepared to receive it and all implementations may ignore it, i.e., it shall not cause a disconnect.

6.1.2.2.5 Class 0 Error-TPDU

When Transport Class 0 has been negotiated, the Error Transport Protocol Data Unit (ER-TPDU) may be used at any time and upon receipt it requires that the recipient disconnect the network connection and, by extension, the Transport connection.

6.1.2.2.6 Unknown CR TPDU parameters

An unknown parameter in any received CR TPDU shall be ignored.

When all of the classes are supported, the preferred class, when initiating a CR TPDU, shall be Class 4.

If a responder receives an alternative class of "none", implicit negotiation is enforced.

6.1.2.2.7 Invalid values of known CR TPDU

Known parameters with valid lengths but with invalid values in a CR TPDU shall be handled as depicted in Table 4.

Table 4/Q.812 – TPDU parameters

Parameter	Action
TSAP-ID	Send DR TPDU
TPDU size	Ignore parameter, use default
Version	Ignore parameter, use default
Checksum	Discard CR TPDU
Alternate protocol classes	Protocol error

6.1.2.2.8 Additional options parameter

Unrecognized or not applicable bits of the "additional options" shall be ignored.

6.1.2.2.9 Network Connection Management Subprotocol (NCMS)

The use of the NCMS, as specified in Annex B of ITU-T Rec. X.224 | ISO/IEC 8073 [4], is optional. Implementations that support the NCMS shall be able to communicate with implementations that do not support the NCMS.

6.1.3 ISO TP0/TCP/IP Profile for use with the IP service

This clause defines the transport protocol profile for use with OSI paradigm when the lower layer IP service defined in ITU-T Rec. Q.811 [45] is used.

– For the top of Layer 4 – STD0035 "ISO Transport Service on top of the TCP (Version 3)" March 1997. (Includes RFC 2126). This document defines how to provide the TP0, ISO Transport Services over TCP.

– For the bottom of Layer 4 – STD0007 "Transmission Control Protocol", September 1981. (Includes RFC 793.)

It should be noted that STD0035 (RFC 2126) implements the ISO TP0 protocol on top of TCP/IP, not on top of the ISO/ITU-T Network protocol. Since the Transport class 0 protocol is used over the TCP/IP connection, it achieves identical functionality as Transport Class 4. Hence, ISO/ITU-T higher level layers (all session, presentation, and application entities) can operate fully without knowledge of the fact that they are running on a TCP/IP internetwork.

6.2 Session layer

6.2.1 Service definition

The session layer conforms to the service definition in ITU-T Rec. X.215 | ISO/IEC 8326.

The default values shall be part of a vendor's offering. That is, unless otherwise specified by the user, the default parameters shall be the initial values supplied. They can be subsequently changed by the user within the specified range.

A conflict in the code values for subsequent number and flow control confirmation exists between ISO and ITU-T. The conflict is expected to be resolved as specified in ITU-T Rec. X.224 | ISO/IEC 8073 [4].

6.2.1.1 Functional units

Two session layer Functional Units (FUs) are required in this Recommendation:

1) Kernel;

2) Duplex.

6.2.2 Protocol specification

The session layer conforms to the protocol definition in ITU-T Rec. X.225 | ISO/IEC 8327-1 [5]. The specific options and parameter values that shall be supported for Telecommunications Systems Management application are specified in ISO/IEC ISP 11183-1 [6].

6.2.2.1 User data

The maximum length of the session user data shall be 10 240 octets. This restriction implies that the Overflow Accept and Connect Data Overflow SPDUs are not required to be supported. "Session-selector" parameter values shall have a maximum length of 16 octets.

6.3 Presentation layer

6.3.1 Service definition

It is mandatory that the presentation layer conform to the services specified in ITU-T Rec. X.216 | ISO/IEC 8822 [7].

6.3.1.1 Functional units

One presentation layer Functional Unit (FU) is required in this Recommendation:

– Kernel.

6.3.2 Protocol specification

It is mandatory that the presentation layer conform to the protocols specified in ITU-T Rec. X.226 | ISO/IEC 8823-1 [8] (normal mode). The specific options and parameter values that shall be supported for Telecommunications system Management application are specified in ISO/IEC ISP 11183-1 [6].

6.3.3 Encoding rules for transfer syntax

The encoding rules defined in ITU-T Rec. X.209 | ISO/IEC 8825 [9] shall be applied to derive the transfer syntax for the Application Protocol Data Units (APDUs). The ASN.1 [10] to [13] OBJECT IDENTIFIER [joint-iso-itu-t asn1 (1) basic-encoding (1)] shall be used as the value for the transfer syntax name. The maximum value of an ASN.1 basic encoding tag that needs to be handled for conformance to this Recommendation is 16 383. This is the largest unsigned integer that can be represented in 14 bits. Hence the identifier octets shall consist of an initial octet and up to two more octets, thus occupying a maximum of three octets. Also, the largest number of octets in the "contents octets" component of an ASN.1 data value encoding that needs to be handled for conformance to this Recommendation is 4 294 967 295. This is the largest unsigned integer that can be represented in 32 bits. Hence in the "long form" encoding, the length octets shall consist of an initial octet and up to four more octets, thus occupying a maximum of five octets. (Note that this restriction does not apply to "indefinite length" encodings.)

6.4 Application layer

The application layer protocol data unit presentation is described by using Abstract Syntax Notation One (ASN.1), as defined in ITU-T Rec. X.208 | ISO/IEC 8824 [14].

6.4.1 Application layer architecture

It is mandatory that the application layer conforms to the architecture for the application layer outlined in ISO/IEC 9545 [15].

The concepts of Application Entity (AE), Application Entity Invocation, Application Service Object (ASO), Control Function (CF) and Application Context will be used to describe the relationship between ROSE, ACSE, CMISE, SMASE.

6.4.2 Association control service element

6.4.2.1 Service definition

The ACSE service description is detailed in ITU-T Rec. X.217 | ISO/IEC 8649 [16]. All of the defined ACSE services (see Table 5) are mandatory. The value of mode parameter of A-ASSOCIATE shall be "normal".

6.4.2.2 Protocol specification

The protocol specification for ACSE shall follow ITU-T Rec. X.227 | ISO/IEC 8650-1 [17]. All five APDUs (see Table 5) specified in the standard are mandatory. The specific options and parameter values that shall be supported for Interactive Class Telecommunications Systems Management application are specified in ISO/IEC ISP 11183-1 [6].

Table 5/Q.812 – ACSE services and associated APDUs

ACSE service	Associated APDUs	Related P-service
A-ASSOCIATE	AARQ, AARE	P-CONNECT
A-RELEASE	RLRQ, RLRE	P-RELEASE
A-ABORT	ABRT	P-U-ABORT
A-P-ABORT	(None)	P-P-ABORT

6.4.2.3 Use of the SACF for association control

The CF is defined to control the interactions among ASEs and/or ASO within the containing ASO in ISO/IEC 9545 [15] with DAM 1.

Thus it controls the association establishment, release and abort with respect to the rules defined in the Application Context available for the association.

Then it allows the joint use of several ASEs on the same association.

6.4.2.4 Abstract Syntax Name

The ACSE abstract syntax name has the ASN.1 type OBJECT IDENTIFIER. The following value shall be used to identify the ACSE abstract-syntax-definition:

```
{
joint-iso-itu-t association-control (2)
abstract-syntax (1) apdu's (0) version (1)

}
```

6.4.3 Remote operations

6.4.3.1 Service definition

The Remote Operations Service Element (ROSE) shall be a mandatory service element. The ROSE service description is detailed in ITU-T Rec. X.219 | ISO/IEC 9072-1 [18]. All of the defined ROSE services (see Table 6) are mandatory.

6.4.3.2 Protocol specification

The protocol specification for ROSE shall follow ITU-T Rec. X.229 | ISO/IEC 9072-2 [19]. All four APDUs specified in the standard (see Table 6) are mandatory. In addition, the ability to support correct origination and reception of the linked-id protocol element is required.

The requirement specified in Table 6 implies association Class 3 in ROSE.

Table 6/Q.812 – ROSE services and associated APDUs

ROSE service	Associated APDUs	Related underlying service
RO-INVOKE	ROIV	P-DATA
RO-RESULT	RORS	P-DATA
RO-ERROR	RORE	P-DATA
RO-REJECT-U	RORJ	P-DATA
RO-REJECT-P	RORJ	P-DATA

6.4.4 Common management information

Network management applications shall use the Common Management Information Service Element (CMISE).

6.4.4.1 Service definition

The CMISE service description is detailed in ITU-T Rec. X.710 | ISO/IEC 9595 [20]. The CMISE services are listed in Table 7.

Multiple object selection, filter, multiple reply and cancel get functional units as defined in ITU-T Rec. X.710 | ISO/IEC 9595 [20] are optional. Their use is application dependent. The negotiation during association establishment to use or not use the functional units shall be supported.

Support of the extended service functional unit defined in ITU-T Rec. X.710 | ISO/IEC 9595 [20] is not required for conformance to this Recommendation and negotiation shall be supported, at association establishment, for its non-use.

Table 7/Q.812 – CMISE services

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

6.4.4.2 Protocol specification

Implementations shall support those operations defined in ITU-T Rec. X.711 | ISO/IEC 9596-1 [21], that are required by specific applications. All mandatory parameters defined in ITU-T Rec. X.711 | ISO/IEC 9596-1 [21] for the required operations are mandatory parameters for this Recommendation. The specific options and parameter values that shall be supported are specified in ISO/IEC ISP 11183-3 [22] for basic Telecommunications Systems Management and in ISO/IEC ISP 11183-2 [23], for enhanced Telecommunications Systems Management.

6.4.4.3 Abstract syntax

The abstract syntax name for CMISE is {joint-iso-ccitt ms(9) cmip(1) abstract syntax(4)}.

6.5 Security support for interactive applications

For X-Interface, the support for authentication and access control security services are mandatory, For Q, the support for these services are optional. The authentication service shall be supported using Authentication Functional Unit specified in ACSE. The actual mechanism(s) to be used for the X-Interface is for further study.

The access control service shall be supported by using the access control parameter defined in CMIP operations. The syntax for this parameter depends on the specific mechanism and is for further study. When specific mechanisms are defined, an additional abstract syntax defining the syntax of the access control shall be included in the Definition Context Set (DCS) for the Presentation Protocol.

7 Upper layer protocol specification for File-oriented Class functions using the OSI paradigm

The profiles for each layer are the same as described in clause 6; this clause only documents the differences required for FTAM support.

7.1 Session layer

7.1.1 Service profile

7.1.1.1 Functional units

Four session layer Functional Units (FUs) are required in this Recommendation:

1) Kernel;
2) Duplex;
3) Minor Synchronize;
4) Resynchronize.

7.1.2 Protocol profile

The specific options and parameter values that shall be supported for file transfer service are specified in ISO/IEC ISP 10607-1 [24].

7.2 Presentation layer

7.2.1 Service definition

It is mandatory that the presentation layer conform to the services specified in ITU-T Rec. X.216 | ISO/IEC 8822 [7].

7.2.1.1 Functional units

One presentation layer Functional Unit (FU) is required in this Recommendation:

– Kernel.

7.2.2 Protocol specification

It is mandatory that the presentation layer conform to the protocols specified in ITU-T Rec. X.226 | ISO/IEC 8823-1 [8] (normal mode). The specific options and parameter values that shall be supported for Telecommunications system Management application are specified in ISO/IEC ISP 11183-1 [6].

7.2.3 Encoding rules for transfer syntax

The encoding rules defined in ITU-T Rec. X.209 | ISO/IEC 8825 [9] shall be applied to derive the transfer syntax for the Application Protocol Data Units (APDUs). The ASN.1 OBJECT IDENTIFIER [joint-iso-itu-t asn1 (1) basic-encoding (1)] shall be used as the value for the transfer syntax name. The maximum value of an ASN.1 basic encoding tag that needs to be handled for conformance to this Recommendation is 16 383. This is the largest unsigned integer that can be represented in 14 bits. Hence the identifier octets shall consist of an initial octet and up to two more octets, thus occupying a maximum of three octets. Also, the largest number of octets in the "contents octets" component of an ASN.1 data value encoding that needs to be handled for conformance to this Recommendation is 4 294 967 295. This is the largest unsigned integer that can be represented in 32 bits. Hence in the "long form" encoding, the length octets shall consist of an initial octet and up to four more octets, thus occupying a maximum of five octets. (Note that this restriction does not apply to "indefinite length" encodings.)

7.3 Application layer profile

7.3.1 Application layer architecture

The description of ACSE and FTAM as part of the application layer architecture is to be provided.

7.3.2 File transfer, access and management

7.3.2.1 Service profile

The mandatory file service class is file transfer class.

In this class the following functional units are mandatory:

– the kernel functional unit;
– both the read and write functional units;
– the limited file management functional unit;
– the grouping functional unit;
– and, in the internal file service, the recovery functional unit and optionally the restart functional unit.

7.3.2.2 Protocol profile

The functional units of the file protocol are equivalent to the functional units of the supported service described above.

The functional units retained and their PDUs associated are listed in Table 8.

This file protocol assumes the session services described in 7.1.1.1 with the following details:

– the recovery or restart functional unit implicates the use of minor synchronize session service;
– the restart functional unit implicates in addition to minor synchronize session service the resynchronize session service.

Table 8/Q.812 – FTAM functional units and PDUs associated

Name	Functional units
F-INITIALIZE request	Kernel
F-INITIALIZE response	Kernel
F-TERMINATE request	Kernel
F-TERMINATE response	Kernel
F-P-ABORT request	Kernel
F-U-ABORT request	Kernel
F-SELECT request	Kernel
F-SELECT response	Kernel
F-DESELECT request	Kernel
F-DESELECT response	Kernel
F-CREATE request	Limited file management
F-CREATE response	Limited file management
F-DELETE request	Limited file management
F-DELETE response	Limited file management
F-READ-ATTRIB request	Limited file management
F-READ-ATTRIB response	Limited file management
F-OPEN request	Read, write
F-OPEN response	Read, write
F-CLOSE request	Read, write
F-CLOSE response	Read, write
F-READ request	Read
F-WRITE request	Write

Table 8/Q.812 – FTAM functional units and PDUs associated

Name	Functional units
F-DATA-END request	Read, write
F-TRANSFER-END request	Read, write
F-TRANSFER-END response	Read, write
F-CANCEL request	Read, write
F-CANCEL response	Read, write
F-BEGIN-GROUP request	Grouping
F-BEGIN-GROUP response	Grouping
F-END-GROUP request	Grouping
F-END-GROUP response	Grouping
F-RECOVER request	Recovery
F-RECOVER response	Recovery
F-RESTART request	Restart
F-RESTART response	Restart

7.3.2.3 Abstract syntax

The abstract syntax names for FTAM are:

{iso standard 8571 abstract syntax(2) ftam-fadu(2)}

{iso standard 8571 abstract syntax(2) ftam-pci(1)}

{iso standard 8571 abstract syntax(2) unstructured-text(3)}

{iso standard 8571 abstract syntax(2) unstructured-binary(4)}

7.3.2.4 Support of document types

The nature of the file structures to be transferred involves the use of the suitable document types.

Three types of file structures are retained:

– unstructured binary files;

– unstructured text files;

– sequentially ordered files (these files are made of a sequence of records without any possibility of having direct access to a given record, each record is made of fields of different types).

So three document types at least are mandatory:

– ISO FTAM unstructured text (FTAM.1);

– ISO FTAM unstructured binary (FTAM.3);

– NBS sequential file (NBS-6).

FTAM.1 and FTAM.3 are allowed by FTAM hierarchical file model defined in ISO 8571-2 [26] as constrained by the unstructured constraint set.

NBS-6 is allowed by FTAM hierarchical file model defined in ISO 8571-2 [26] as constrained by the sequential flat constraint set.

7.4 Security support for FTAM services

For X-Interface, the support for authentication service is mandatory. For Q, the support for these services is optional. The authentication service shall be supported using Authentication Functional Unit specified in ACSE. The actual mechanism(s) to be used for the X-Interface is for further study.

Security support for FTAM services in TMN is for further study.

8 Upper layer protocol specification for Directory services using the OSI paradigm

The profiles for each layer are the same as described in clause 6; this clause only documents the differences required for directory support.

8.1 Session layer

8.1.1 Service definition

This layer conforms to the service definition in ITU-T Rec. X.215 | ISO/IEC 8326.

8.1.1.1 Functional units

Two session layer FUs are required in this Recommendation:

a) Kernel;

b) Duplex.

8.1.2 Protocol specification

The session layer conforms to the protocol definition in ITU-T Rec. X.225 | ISO/IEC 8327-1 [5].

8.1.3 User data

DUAs shall be capable of sending request APDUs of any size up to 32 767 (32k – 1) octets in length. DSAs shall be capable of accepting and processing operation request APDUs of any size up to 32 767 octets in length. DSAs shall be capable of sending response APDUs of any size up to 262 143 (256k – 1) octets in length. DSAs shall be capable of accepting and processing response APDUs of any size up to 262 143 octets in length and shall be capable of sending request APDUs of any size up to 32 767 octets in length.

8.2 Presentation layer

8.2.1 Service definition

The presentation-service is defined in ITU-T Rec. X.216 | ISO/IEC 8822 [7].

The ACSE is the sole user of the P-CONNECT, P-RELEASE, P-U-ABORT and P-P-ABORT services of the presentation-service.

The ROSE is the sole user of the P-DATA service of the presentation service.

Presentation default context, context restoration, and context management are not used.

8.2.2 Protocol specification

It is mandatory that the presentation layer conform to the protocols specified in ITU-T Rec. X.226 | ISO/IEC 8823-1 [8] (normal mode).

8.3 Application layer

8.3.1 Application layer architecture

It is mandatory that the application layer conforms to the architecture for the application layer outlined in ITU-T Recs X.500.x | ISO/IEC 9594 [29] to [36].

8.3.2 Directory protocol abstract syntaxes

The ASN.1 type from which the values of the abstract syntaxes are derived is specified using the parameterized types of ROS {DAP-InvokeIDSet | DAP-Invokable | DAP-Returnable | DSP-InvokeIDSet | DSP-Invokable | DSP-Returnable}, Bind {dSABind | directoryBind}, and Unbind {dSAUnbind | directoryUnbind} which are defined in ITU-T Rec. X.880 | ISO/IEC 13712-1 [37].

The DAP abstract syntax is called the directoryAccessAbstractSyntax. The DSP abstract syntax is called the directorySystemAbstractSyntax.

8.3.3 Directory application contexts

The DAP application context is called the directoryAccessAC. The DSP application context is called the directorySystemAC.

8.3.4 Association control service element

The abstract-syntax of ACSE, acse-abstract-syntax is required for DAP and DSP.

The ACSE supports the establishment, release and abort of an application-association between a pair of AEs. Associations between a DUA and a DSA may be established only by the DUA. Only the initiator of an established association can release it.

8.3.4.1 Service definition

The ACSE service description is detailed in ITU-T Rec. X.217 | ISO/IEC 8649 [16].

The RO-BIND and RO-UNBIND services are the sole users of the A-ASSOCIATE and A-RELEASE services of the ACSE. The application-process is the user of the A-ABORT and A-P-ABORT services of the ACSE.

8.3.4.2 Protocol specification

The protocol specification for ACSE shall follow ITU-T Rec. X.227 | ISO/IEC 8650-1 [17].

8.3.5 Remote operations

8.3.5.1 Service definition

ROSE shall be a mandatory service element. The ROSE service description is detailed in ITU-T Rec. X.881 | ISO/IEC 13712-2 [38].

The Directory ASEs are users of the RO-INVOKE, RO-RESULT, RO-ERROR, RO-REJECT-U, and RO-REJECT-P services of the ROSE.

8.3.5.2 Protocol specification

The DAP and DSP are the Directory protocols used to provide communications between a pair of application processes.

8.4 Security support for directory services

ITU-T Rec. X.509 | ISO/IEC 9594-8 [36] defines a framework for the provision of authentication services by the Directory to its users. Security support for Directory services in TMN is for further study.

The upper layer protocols to be used for store and forward services (e.g., for exchange of EDI formatted information) is for further study.

This Recommendation specifies partial support for security requirements across the Q and X interfaces. To support security services such as data integrity, confidentiality and non-repudiation and management of security information (such as key management procedures and protocols), the use of Generic Upper Layer Security Recommendations (X.830 series of ITU-T Recommendations) will be required. Guidelines for using GULS in applications are specified in Annex A/X.830 | ISO/IEC 11586-1 [39]. Details for using GULS in both interactive and file transfer classes of TMN applications are for further study.

9 Conformance for the OSI paradigm

Requirements for items not specifically referenced in this Recommendation shall be per ISPs as identified below:

– Transport:
  - For CLNS1 (see ITU-T Rec. Q.811 [45]) the transport layer shall conform to ISO/IEC ISP 10608-1 [60].
  - For CLNS2 (see ITU-T Rec. Q.811 [45]) the transport layer shall conform to ISO/IEC ISP 10608-1 [60].
  - For CLNS3 (see ITU-T Rec. Q.811 [45]) the transport layer shall conform to ISO/IEC ISP 10608-1 [60].
  - For CONS1 (see ITU-T Rec. Q.811 [45]) the transport layer shall conform to ISO/IEC ISP 10609-1 [61] as modified by Table II.1.
  - For CONS6 the transport layer shall conform to ISO/IEC ISP 10609-1 [61].
  - For ISO TP0/TCP/IP stack transport layer shall conform to class 0 of RFC 2126.

– Session, Presentation and ACSE layers for Interactive Class services shall conform to ISO/IEC ISP 11183-1 [6].

– Session, Presentation and ACSE for File-oriented Class services shall conform to ISO/IEC ISP 10607-1 [24].

– CMIP utilized in Interactive Class services profile shall conform to ISO/IEC ISP 11183-3 [22] for basic services and to ISO/IEC ISP 11183-2 [23] for enhanced services. Applications may override the APDU size of 10K specified in AOM-12 if larger size is required.

– FTAM profile shall correspond to ISO/IEC ISP 10607-3 [40].

10 Protocol profile for CORBA-based services

10.1 Scope of CORBA protocol profile

TMN applications specified using IDL shall interoperate according to the provisions of this CORBA protocol profile.

10.2 Overview of profile for CORBA-based services

Figure 4 illustrates the protocol stack for the profile for CORBA-based services.

TMN services which have object-oriented interfaces specified using ODP IDL (ITU-T Rec. X.920) may be accessed through use of this profile.

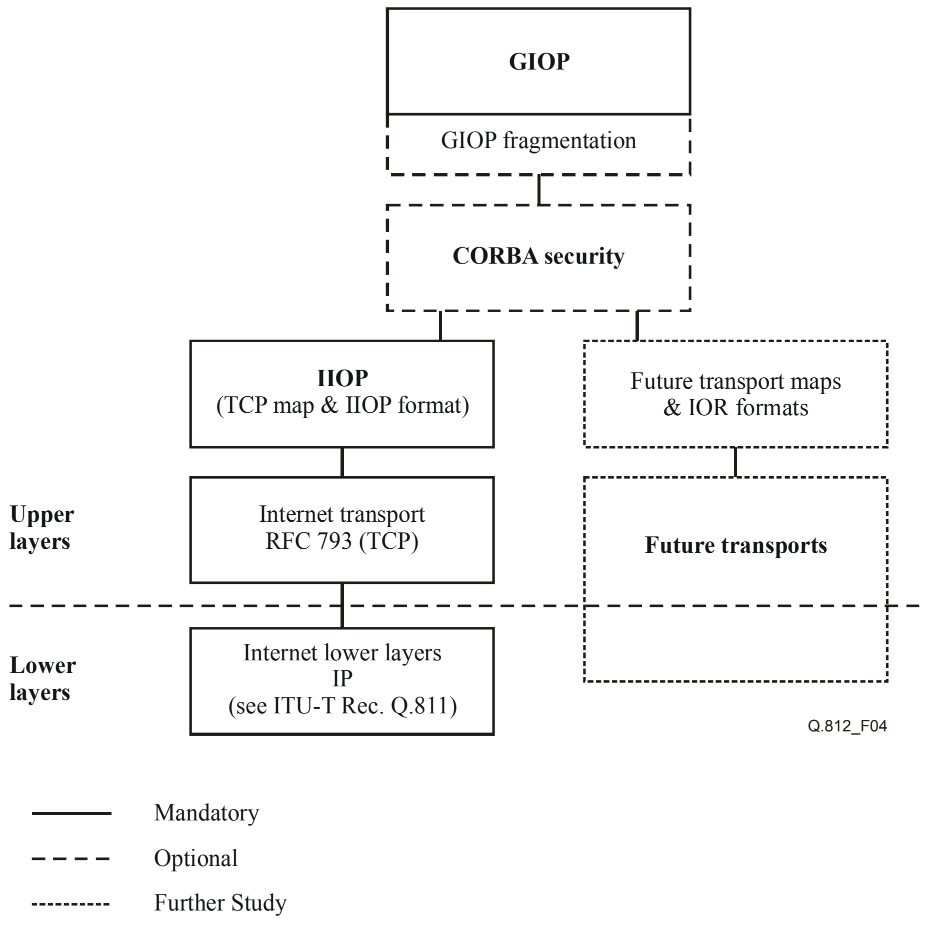


Figure 4/Q.812 – GIOP upper layer CORBA-based services protocol stack

To support CSI level 1 and above, the SECIOP protocol must be used within this profile. For support of CSI level 0, CORBA Security SSL Interoperability may be used as an alternative to SECIOP within this profile. Systems which support CORBA security must also support the IIOR profile version 1.1 format.

If application fragmentation is required, then GIOP version 1.1 or later must be used within this profile.

Mappings of GIOP onto transports other than TCP is for further study.

NOTE – GIOP mappings to transport profiles require the specification of an Interoperable Object Reference (IOR) profile format, associated with that transport profile, as well as the specification of how binding services of the transport profile are used.

10.3 Service definition

Services which use CORBA must have object-oriented interfaces specified using OMG IDL (ITU-T Rec. X.920).

NOTE – CORBA-based systems may use standard programming language bindings to access CORBA objects.

10.4 GIOP protocol specification

GIOP versions 1.0, 1.1 and 1.2 are to be implemented as specified in [CORBA GIOP Specification]. All systems which act as CORBA servers must support at least GIOP 1.0.

Servers which support GIOP versions 1.1 or 1.2 must also support processing messages with all previous GIOP versions.

10.5 Secure IOP protocol specification

All systems which require use of CORBA security services must support GIOP version 1.1 or later.

All systems which require use of CORBA security must support either the "Secure IOP protocol", or "CORBASecurity SSL Interoperability", as defined in [CORBA Security Service Specification].

10.6 IIOP protocol specification

For interoperability, all CORBA systems shall support the IIOP mapping protocol of GIOP onto TCP/IP lower layer services, as specified in [CORBA GIOP Specification].

Servers shall indicate their support of GIOP through publishing Interoperable Object References (IOR) which include an Internet IOR (IIOR) profile with IIOP profile version set to the highest level of GIOP protocol version supported by the system acting as server. The IIOR profile format is as specified in [CORBA GIOP Specification].

10.7 TCP/IP protocol profile for use with IIOP

IIOP is designed to be used with TCP/IP-based lower layer protocols.

This clause defines a protocol profile for use as TMN lower layer protocols for CORBA-based systems using IIOP. This profile is based on the use of Internet protocols defined by the Internet Engineering Task Force (IETF). The way these documents can be referenced in this Recommendation is for further study. The protocol stack uses the following:

- For Layer 4 – STD0007 "Transmission Control Protocol", September 1981. (Includes RFC 793.)
- For Layer 3 and below the IP protocol profile specified in ITU-T Rec. Q.811 [45] is used.

Other lower layer protocol mappings for GIOP are for further study.

11 Protocol Profile for EDI/EDIFACT based services

11.1 Scope of EDI/EDIFACT protocol profile

TMN applications that have X interface definitions for use at the service management layer shall interoperate according to the provisions of this protocol profile. This Recommendation defines the profile for the Electronic Communication Interactive Agent (IA) and associated layers of functionality. The protocol for the IA itself is found in ITU-T Rec. Q.814. The IA peer-to-peer interface will support near real-time bidirectional data transfer between peer entities.

The overall profile described in this clause is modelled after the seven-layer open systems interconnection (OSI) model, that is, the profile layers herein are described in terms of the transport layer (4), the session layer (5), the presentation layer (6), and the application layer (7).

The IA described herein provides layer five (5) services. The other layers, four, six, and seven, provide functionality that interacts directly or indirectly with the IA. This profile describes the interaction and responsibilities of each of these four layers.

11.2 Layer summary

Refer to Figure 5 in the following discussions.

11.3 TCP/IP protocol profile for use with IA

IA is designed to be used with TCP/IP-based lower layer protocols.

This clause defines a protocol profile for use as TMN lower layer protocols for IA. This profile is based on the use of Internet protocols defined by the Internet Engineering Task Force (IETF). The way these documents can be referenced in this Recommendation is for further study. The protocol stack uses the following:

- For Layer 4 – STD0007 "Transmission Control Protocol", Postel [J.] September 1981. (Includes RFC 793.)
- For Layer 3 and below the IP protocol profile specified in ITU-T Rec. Q.811 [45] is used.

11.4 TLS protocol profile for use with IA

Layer four provides transport layer security, and transport services utilizing the Transmission Control Protocol (TCP) (see above).

The transport mechanism specified by the Interactive Agent (IA) requires an individual TLS session. This session either may be persistent or may be established or resumed for each message. The communication is essentially one way, from the client to the server. The IA status message is a mechanism that allows peer entities to exchange errors and other types of flow control information. Specific message codes may be defined by the peer entities outside the scope of this Recommendation. TLS provides secure handshake and transfer between peer TLS entities. TLS also provides for data flow integrity, peer entity authentication, and, optionally, privacy.

11.5 IA profile

The IA performs the session layer functionality. The IA supports the exchange of Electronic Data Interchange (EDIFACT/ASC X12 EDI/general string) transactions between peer entities. The IA supports this interchange over Transport Layer Security (TLS). The session layer functions provided by the IA include the establishment, management and closing of communications sessions between peer entities. The IA also performs the conversion of EDIFACT/ASC X12 EDI recipient names to network addresses and manages the TLS session. At the conclusion of a session, the IA will determine whether to close a session or to suspend it in a state whereby it can be *resumed*.

The IA Service Agreement interface is defined at the boundary between the IA and its direct user.

The protocol supporting the exchange of IA messages is referred to as the interactive agent transfer protocol (IATP), and is defined in ITU-T Rec. Q.814.

11.6 Security module for whole message protection profile

This Security Module functionality is optional depending on the security needs of the transaction. However, if whole message security services are needed, the procedures of ITU-T Rec. Q.815 shall be followed. Secure messages are transferred between security modules providing both non-repudiation of origin and receipt, and message integrity.

The Security Module generates/validates the appropriate security fields and performs the required encoding/decoding depending on whether it is sending/receiving, respectively.

Message flow between the EDI Translator and the IA may or may not require security services. The case where security enhancements are applied to a message is depicted in Figure 5.

The security module is not sensitive to the content of messages coming from the EDI Translator or the IA.

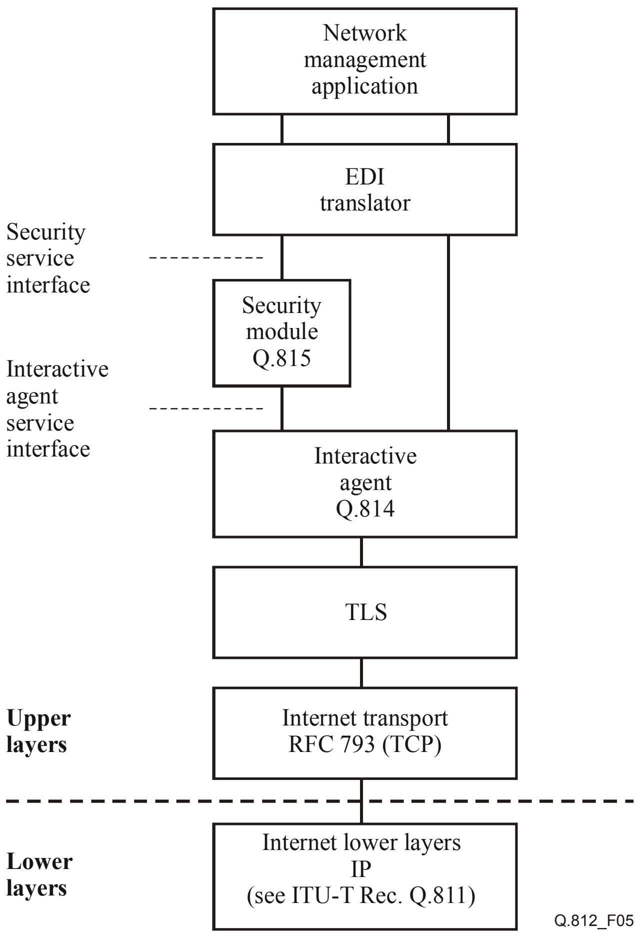

Network management application
EDI translator
Security service interface
Security module Q.815
Interactive agent service interface
Interactive agent Q.814
TLS
Upper layers
Internet transport RFC 793 (TCP)
Lower layers
Internet lower layers IP (see ITU-T Rec. Q.811)
Q.812_F05

Figure 5/Q.812 – Upper layer EDI/EDIFACT based services protocol stack

11.7 EDI/EDIFACT translator protocol

EDI/EDIFACT protocol contains the user application that uses the services of both the Interactive Agent and, optionally, the Security Module.

An EDIFACT/ASC X12 EDI translator/gateway is an application service that provides a combination of data format translations and data interchange functions for electronic transaction message data.

An EDIFACT/ASC X12 EDI translator/gateway exchanges transaction data to and from network management applications via intermediate data formats. It translates this data to and from externally defined EDIFACT/ASC X12 EDI data formats using translation maps.

12 Protocol profile for the SNMP paradigm

The SNMP paradigm is shown in Figure 6. Version 3 of the Internet Management Framework is described in IETF RFC 3410. This framework consists of a data definition language [57], definitions of management information, a protocol definition [54], security [52] and [53] and administration [51]. The protocol is primarily run over UDP [55], but can alternatively be run over TCP [58]. Coexistence with previous versions of SNMP is described in [56].

Where SNMP is deployed, Version 3 is the preferred version. Previous versions of the Internet Management Framework may be secured using IPSec [59].

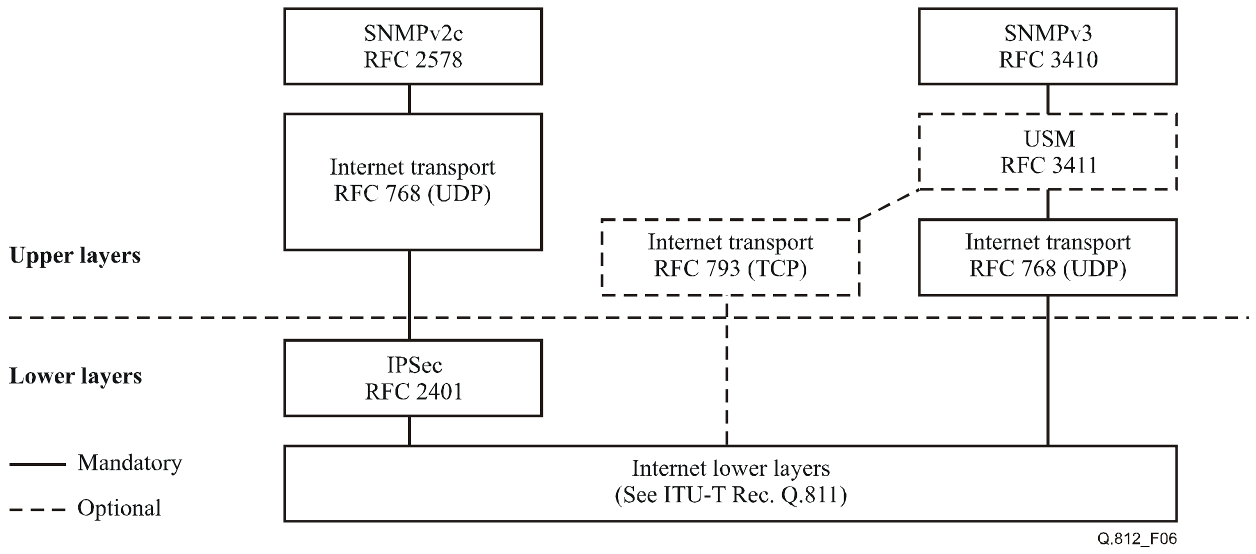


Figure 6/Q.812 – Protocol profile for the SNMP paradigm

13 Protocol profile for the Telecommunications Markup Language (tML) paradigm

A protocol profile to support use of tML [64] encoded management information is for further study.

Appendix I

Guidance on using allomorphic management

I.1 Introduction

This appendix seeks to provide guidance to the developers of CMIP managers and agents in the use of allomorphism. Allomorphism is a powerful concept that is of increasing value as TMNs are implemented. Allomorphism can be used to address the issue of how to add new capabilities to existing TMN manager and agent implementations. As requirements evolve and models are extended to satisfy those requirements, software in the managing system that takes advantage of allomorphism can be written in such a way that it does not require to be rewritten until the new features in the model are needed.

This appendix attempts to clarify how to cope with allomorphic behaviour in implementations of both manager and agent systems. It clarifies the description of allomorphism found in ITU-T Rec. X.720 | ISO/IEC 10165-1. In particular, managers must be aware of allomorphism to benefit from it. Even if a manager does not plan to use allomorphism, it should have a minimum ability to interface with agents that do implement allomorphism. For example, the manager must support the allomorphs attribute and have the ability to construct filters using allomorphs versus the actual class in the objectClass attribute.

This appendix discusses the issues related to allomorphism for each CMIP operation from both the manager and agent perspective. It then discusses the issues related to CMIP notifications, again from both the manager and agent perspective. It then provides some protocol stack and implementation considerations. It concludes by answering some frequently asked questions about allomorphism.

In the following discussion, the phrase "if agent supports allomorphism" is used. This should be equated to "if agent supports allomorphism for a specific instance" because it is possible that an agent may support revisions for some object classes and not others. Strictly speaking, even two instances of a class may be different in the support for allomorphism, however, this is considered as an extreme and rare implementation for this discussion. The same is also true for manager side where a specific release of a manager system may recognize multiple definitions for some basic classes and not for others. The decisions to include the different versions are dictated by business objectives (which are out the scope of this appendix).

I.1.1 Overview

Allomorphism is the ability of a managed object that is an instance of a given managed object class to be managed as a member of one or more other managed object classes. Allomorphism allows instances of one managed object class – referred to as the extended class to represent instances of another managed object class – the allomorphic class.

When an extended class is instantiated, the actual class (see ITU-T Rec. X.720 | ISO/IEC 10165-1) of the object stored in the objectClass attribute is the extended class. It is extended with regard to another managed object class, its compatible managed object class. The actual class is that class of which a managed object is an instance. An allomorphic class of a managed object is a managed object class other than the managed object's actual class; however, it can be managed as an instance of that class. A managed object may be allomorphic to one or more of the compatible classes (i.e., instances of the extended class can be managed as instances of compatible managed object class). In other words, the terms "allomorphic class" and "compatible managed object class" can be used as synonyms. When an agent creates a managed object that supports allomorphism, the allomorphicPackage (defined as a Conditional Package in the top class in ITU-T Rec. X.721 | ISO/IEC 10165-2) is included. The package contains the allomorphs attribute. This GET-Only attribute is set-valued and contains the object identifiers of the classes that this object can represent (is allomorphic with). The objectClass attribute has a value of the actual class used in creating this instance.

The basic idea behind allomorphism is that the extended class supports all the capabilities of the classes it is allomorphic with. It may also support additional capabilities. The extended class may be a subclass of the classes it is allomorphic with but this is not required. In all respects, the extended class behaves as the class it actually is. This may mean that the manager could receive information that is not in the allomorphic class. For example, if the extended class has new attributes, a Get all operation from the manager will return values for these. Dealing with this type of issue requires the manager to be aware that allomorphic management is being used. ITU-T Rec. X.724 | ISO/IEC 10165-6 defined Managed Object Conformance tables for use by both agent and manager implementations. Use of these tables to identify the list of allomorphs if supported is recommended to determine the levels of interoperability.

The various interactions between manager and agents using allomorphic management are discussed in the following clauses. ITU-T Rec. X.720 | ISO/IEC 10165-1 also discusses limited interoperability when compatible rules are not completely satisfied. This appendix addresses only the scenarios where compatible rules defined according to ITU-T Rec. X.720 | ISO/IEC 10165-1 to support allomorphism are met (see 5.2.3.2/X.720 | ISO/IEC 10165-1).

I.2 CMIP operations

This clause discusses allomorphism in relation to the CMIP Operations, m-Create, m-Get, m-Set, m-Action, and m-Delete. No impact of allomorphism on the m-Cancel-Get operation has been identified.

I.2.1 Creating managed objects

A managed object is created either by the manager issuing an explicit create request on the interface or by automatic create within the agent system. Each case is discussed below. Careful selection of the naming attribute and structure is required. The naming issues are discussed below in a separate subclause.

I.2.1.1 Explicit creation – Manager role

Case 1: The manager issues a CMIP create request providing in the managed object class a value and a set of attribute values appropriate for that class. The manager supplies the actual name of the new object. The resulting action in the agent is one of those defined in I.2.1.2 Explicit creation – Agent role, cases a, b, c. If the response is as defined in case c, the manager must be able to ignore unknown attributes included as a result of creating an extended class.

Case 2: The manager issues a CMIP create request providing in the managed object class a value and a set of attribute values appropriate for that class. The manager supplies the name binding attribute value. The resulting action in the agent is one of those defined in I.2.1.2 Explicit creation – Agent role, cases a, b, d. If the response is as defined in case d, the manager must be able to ignore unknown attributes included as a result of creating an extended class.

Case 3: The manager specifies the class without a specific name in the instance field or a value for the name binding attribute. The resulting action in the agent is one of those defined in I.2.1.2 Explicit creation – Agent role, cases a, b, e. If the response is as defined in case e, the manager must be able to ignore unknown attributes included as a result of creating an extended class.

Case 4: The manager specifies a class and requests it to be a copy of another object, i.e., with reference object. Depending on the value of the class, any of the above three cases are possible.

I.2.1.2 Explicit creation – Agent role

Case a: Agent recognizes the managed object class in the request and supports it as actual class. In this case the requested managed object class is created and allomorphism is not involved. The create succeeds or fails depending on the conditions associated with the behaviour and the attribute values supplied by the manager. The agent uses either the name supplied in case 1 above or assigns a name (using either the name binding rule in case 2 or internally generated based on the schema definition).

Case b: Agent does not support the requested class either as an actual class supported or as an allomorph. The managed object class provided in the request is not recognized. The create request is rejected with the error "no such object class". This is a normal failure to create an unknown class.

Case c: Agent supports allomorphism and it creates a class that is an extended class with the name supplied by the manager. This assumes the name provided by the manager follows the structure rules (name binding) for the extended class [same or extended superior class, same Relative Distinguished Name (RDN)]. If this condition is not met, then the create will fail. If the create succeeds, the agent responds with the value for the actual class in the managed object class field and all the attributes appropriate for the actual class. The object is created according to the behaviour of the extended class. If the agent system is providing the interoperability (see 5.2.3.1/X.720 | ISO/IEC 10165-1), the agent includes the attribute allomorphs with the value of the class provided in the create request. If manager system is providing the interoperability (see 5.2.3.2/X.720 | ISO/IEC 10165-1), the manager may query allomorphs attribute and determine that requested object class is allomorphic to the actual class.

Case d: Agent supports allomorphism and it creates a class that is an extended class with the name binding supplied by the manager. This assumes that the name binding supplied is valid for the extended class. This is often true if the extended class is a subclass and the GDMO includes for name binding "WITH SUBCLASSES" clause. The response when create succeeds is the same as for case c. If the name binding is not valid for the extended class (for example, additional behaviour may be included in a different name binding for the extended class even if the naming attribute and structure are the same) the create will fail. Note that if the manager receives invalid value error for the name binding attribute without further information, this will not help in resolving the problem.

From the manager's perspective the request is for a class and the name binding is valid. Because the extended class requires a different behaviour, the agent cannot use that name binding. This is why the recommendation is not to supply the name binding attribute in the create request.

Case e: The agent supports allomorphism and creates a class with an appropriate name selected by the agent (for case 3 of I.2.1.1: Explicit creation – Manager role). The assigned name may or may not be understood by the manager depending on chosen name binding definition. Other information in the response is the same as in case c.

For all the above cases, when the manager does not supply a value for an attribute and a default exists, the value chosen for that attribute is according to the created object class. In cases c, d, and e this can result in the manager receiving values for attributes that exist only in the extended class. The manager may not understand these types and should be able to ignore these types without disrupting the association. Note that the manager may wish to take additional management action to record that it is encountering allomorphic agents (e.g., log information not understood).

In addition to default values of attributes, constraints imposed on initial value, permitted and required values may have differences between the allomorphic class and the extended class. See I.4 for further discussions.

I.2.1.3 Summary

The table below summarizes the various cases for explicit creation discussed from manager role and agent role depending on whether allomorphism is supported or not.

Manager role	**Agent role with allomorphism**	
	Supported	Not supported
Case 1	Cases a, b, c	Cases a, b
Case 2	Cases a, b, d	Cases a, b
Case 3	Cases a, b, e	Cases a, b
Case 4	Cases a, b, c, d, e	Cases a, b

I.2.1.4 Automatic creation – Agent role

The agent may create a managed object internally and inform the managers of the creation by the object creation notification.

Case 1: Assume that the agent has implemented the extended class and allomorphic behaviour. In addition to all the attributes appropriate for the created class, the allomorphs attribute containing all the allomorphic (compatible) object classes is included in the created managed object. The agent sends a create notification for the new object.

In this case, the agent then sends a notification to all the managers using the object creation notification containing the actual class. It includes all the attributes of the class referenced in the managed object class field of the create notification. This includes the allomorphs attribute.

Note that the default values used are consistent with the created class.

Case 2: The agent only implements the extended class and does not exhibit allomorphism for the compatible classes. In this case the object creation notification contains only the information pertaining to the extended class and the allomorphs attribute is not present. For such an environment, it is recommended that the managing system provide for interoperability independent of whether the additional features are required or not. Rejecting the notification will not be useful for a practical TMN environment.

I.2.1.5 Automatic creation – Manager role

The manager receives the object creation notification with the allomorphs attribute and an extended class in the managed object class field. Depending on whether the manager supports allomorphism or not, the cases described in a to d are applicable for case 1 above.

Case a: The manager knows about the extended class and the allomorphs attribute is not needed because the manager does not have to perform allomorphic management. All the characteristics associated with recognizing or not of the attribute identifiers and values are the same when no allomorphism is used.

Case b: The manager does not recognize the managed class value in the notification. If the manager understands the allomorphs attribute identifier, then the manager, before ignoring the notification as unrecognized information, should examine the allomorphs attribute value to determine if it can manage using one of the values in this attribute. This implies that at least one object class value in the allomorph attribute is recognized by the manager. The manager must ignore all the attributes not recognized as belonging to the allomorphic class.

Case c: The manager does not recognize the managed object class value and it has not implemented the ability to recognize the allomorphs attribute identifier. In this case the manager cannot manage this auto-created object. If the creation is sent with a confirmation, an error response may be provided by the manager to say unknown object.

Case d: The manager does not recognize the managed object class value in the notification. The manager understands the allomorphs attribute; however, it does not recognize any of the classes in the allomorphs attribute. In this case the result is the same as case c. Managing the auto-created object by that manager is not possible.

The agent sends a notification using the actual class without the allomorphs attribute (agent does not allomorphism). This corresponds to case a above.

Case e: The manager understands the extended class and the behaviour is as in case a above. The absence of allomorphs attribute is not relevant and the manager can manage the auto-created object.

Case f: The manager does not recognize the extended class. The manager will not be able to manage this object, given that the allomorphs attribute is not supported by the agent. This is true irrespective of whether the manager supports allomorphism or not.

I.2.1.6 Summary for AutoCreation

The table below summarizes the relationship of the manager and agent cases.

Agent case	Manager case
1	a, b, c, d
2	e, f

I.2.2 Get operation

The Get operation may be issued with an explicit set of attribute identifiers, an empty list or a missing list. The two cases (empty list and missing list are treated the same) are further separated in terms of whether the managed object class in the request is the actual class or allomorphic class for the agent. Note that there is a special object identifier (42) that the manager may use in the request to refer to the actual class of the managed object without having to specify the actual class.

I.2.2.1 Manager role

Case 1: Manager issues a get request with a class that is not the actual class of the object but one of the allomorphs supported by the agent. The list of attribute identifiers included is appropriate for this allomorphic class. The response received is according to one of cases a, b, c below. If interoperability is supported by the manager, then receiving a successful response in case b with a different class will be recognized as a valid response. Otherwise, the manager will reject the response (a ROSE reject and not from CMIP).

Case 2: Manager issues a get request with a class which is either the actual class or the special object identifier which implies the actual class. The list of attributes requested may or may not be appropriate for that class. This is because of either conditionally or by using the special object identifier, the manager may include attributes that are not available for the implemented class. The response received is according to case d below.

Case 3: Manager requests specifying a class, name of a managed object and no attribute list. The response received depends on the value in the class, whether allomorphism is supported by agent and the method of interoperability. The response received is one of cases e, f, g given below. If the response in f or g contains attributes, the manager does not understand, they are ignored by the manager.

Case 4: Manager requests specifying either a class supported by the agent and name of a managed object or the special object identifier discussed above and a name. It does not include the attribute list. The response received is according to case h below. The manager should ignore attributes that are not recognized when the class in the response is different from what the manager recognizes (as a result of using the special object identifier).

I.2.2.2 Agent role

Case a: Agent supports allomorphism and recognizes the name of the managed object. The managed object class in the request matches one of the values in the allomorphs attribute. The agent responds with the values of the attributes requested and the class as in the request may or may not be included. If the class is included, it is recommended that the value of the actual class be used in the response. This approach provides for uniform and consistent response irrespective of single object request or multiple object request using scoping. It is also expected that the managed object class parameter in a CMIP response corresponds to the value of the objectClass attribute.

Case b: Agent does not support allomorphism but recognizes the name of the managed object. The agent may either return an error ("no such object class" or "class instance conflict") or respond with the values of the attribute (agent provided interoperability). The latter behaviour is recommended. The class field may be left empty or populated with the actual class. If both class and name are not recognized, an error response is generated (no such object instance or no such object class).

Case c: Agent does not support allomorphism and does not understand the value of the class or the name in the request. An error "no such object class or no such object instance" is returned.

Case d: The agent recognizes the class and the name irrespective of whether it supports allomorphism or not (this corresponds to case 2 where the manager requests using a class that is a compatible class and not the extended class even though the manager knows about the extended class or the simple case where both the manager and agent know about only one class). Agent responds with attribute values (includes errors if the attributes requested are not suitable for that class or have not been implemented as a result of conditionality). The class field may be omitted or the actual class is included.

Case e: Agent recognizes the class and name in the request (irrespective of allomorphism is supported or not) and returns all the attribute identifiers and values for that object. Class and name may be omitted in the response.

Case f: Agent supports allomorphism. The value of the class in the request does not match the value of the actual class even though the name corresponds to one of the objects contained in the system. The class corresponds to a value in the allomorphs attribute. If the agent provides interoperability, the agent responds with only the value of the attributes appropriate for the requested class. If the value of the class is included in the response (it is not required to include either the class or name), then the actual class is used (see explanation above on the same topic). If the manager provides interoperability, the agent returns all of the attributes included in the object. If the value of the class in the request does not correspond to any of the values in the allomorphs attribute, then the agent returns an error "no such object class" to the manager. Note that in order to provide for interoperability, it is recommended that both agent and manager provide some capabilities: the agent supporting allomorphism and the manager ignoring unknown information.

Case g: Agent does not support allomorphism. The requested class is not recognized but an object with that name is available. The agent may either respond with an error "no such object class" or all the attributes relevant to that class. Even though the response is not required to include the class and instance for single object request, it is recommended that the actual class and name be included in this case. This provides the manager with the information that the actual class is an extended class of the compatible class the manager understands.

Case h: Requested class corresponds to the actual class in the agent or the actual class is used because the request contained the special object identifier value. Irrespective of whether allomorphism is supported or not, all the attribute values corresponding to the implemented actual class for that object are returned assuming that the agent recognizes the name. The class and name fields may or may not be in the response. It is, however, recommended that the actual class and name be included in this case when the special object identifier is used in the request. If the name is not recognized, then "no such object instance" error is returned.

I.2.2.3 Summary GET operation

The table below summarizes the relationship of the manager and agent cases.

Manager case	Agent case
1	a, b, c
2	d
3	e, f, g
4	h

I.2.3 Set operation

The Set operation may be issued with different operators. The replace may be specified either with a specific value or "set to default". The default value for an attribute may depend on the actual class. An attribute may be specified in one class with a set of permitted values and a set of required values. The required values must be a subset or an equal set of the permitted values. The extended class shall not increase the permitted values but may remove some as long as these are not in the required set of values (see Figure I.1). Not supporting a permitted value is permitted either in the extended or compatible class as long as this value is not in the required list. Thus, with allomorphic management, when a permitted value for the allomorph is given and this is not included in the extended class, it can be rejected without violating allomorphic behaviour (guaranteed to support all values in the compatible class; however, if the value is not in the permitted set of compatible class, this will not be supported as the list cannot be extended).

I.2.3.1 Manager role

Case 1: Manager issues a set request with a class, name and the value(s) for the attributes (replace/add or remove operator). The class of the object is not the actual class in the agent but a compatible class. If a response is received (only if the request was confirmed), one of cases a, b, c below is valid. If interoperability is supported by the manager, then receiving a successful response in case b with a different class will be recognized as a valid response. Otherwise, the manager will reject the response (a ROSE reject and not from CMIP).

Case 2: Manager issues a set request with a class which is either the actual class or the special object identifier which implies the actual class. The value(s) for the attributes (replace/add or remove operator) may or may not be appropriate for that instance. This is because of conditionality or by using the special object identifier, the manager may be providing values appropriate for the compatible class. If a response is received (only if the request was confirmed), case d below is valid.

Case 3: Manager requests specifying a class, name and a replace with default for one or more attributes. The response (if received) depends on the value in the class, whether allomorphism is supported by agent and the method of interoperability. It is one of e, f or g below. The response received may indicate default values for the attributes that is different from those associated with the requested class. The manager should recognize these values as the result of actual class in the agent being different.

Case 4: Manager requests specifying either a class supported by the agent and name or the special object identifier discussed above and name and a replace with default for one or more attributes. The response, if received, is according to case h below.

I.2.3.2 Agent role

Case a: Agent supports allomorphism and recognizes the name of the managed object. The managed object class in the request matches one of the values in the allomorphs attribute. The agent performs the modification for those attributes (assuming the attribute exists and the value provided is valid). If the request is confirmed, then the agent responds with either an acknowledgment or the modified values. In the latter case, the value for the class field (if present) is the actual class (see the rationale above for Get). If the attributes or values provided in the request are invalid, then an error or partial success (set list error) is returned.

Case b: Agent does not support allomorphism but recognizes the name of the managed object. The agent may not perform the requested modification based on not recognizing the class. If the request was unconfirmed, then the manager has no knowledge of the result unless it issues a get operation. Consider the case where a response is required. Depending on whether the agent performed the operation (successfully or otherwise), it may return one of the following: an error ("no such object class" or "class instance conflict"); confirmation indicating success, error with partial success. The class and name fields are not required to be in the response. If present, it is recommended to include the actual class to inform the manager of the implemented class. If both class and name are not recognized, an error response is generated (no such object instance or no such object class).

Case c: Agent does not support allomorphism and does not understand the name in the request. An error "no such object class" or "no such object instance" is returned.

Case d: Agent recognizes the class and the name irrespective of whether it supports allomorphism or not (this corresponds to case 2 where the manager requests using a class that is a compatible class and not the extended class even though the manager knows about the extended class or the simple case where both the manager and agent know about only one class). The agent performs the modification for those attributes (assuming the attribute exists and the value provided is valid). If the request is confirmed, then agent responds with either an acknowledgment or the modified values. In the latter case, the class and name fields are not required to be in the response. If present, the value for the class field is the actual class (same rationale as in Get). If the attributes or values provided in the request are invalid, then an error or partial success (set list error) is returned.

Case e: Agent recognizes the class and name in the request (irrespective of allomorphism is supported or not) and performs the operation. The response, if required, may be an acknowledgment, an error because there is no default defined or the modified value (the default specified for that class).

Case f: Agent supports allomorphism. The value of the class in the request does not match the value of the actual class even though the name corresponds to one of the objects contained in the system. The class corresponds to a value in the allomorphs attribute. If the agent provides interoperability, the agent either performs the modification according to the default value for the actual class or detects an error (e.g., no default value defined for some of the attributes). It responds with either an acknowledgment or the assigned default value or an error with partial success. It is not required to include the class and name fields in the response. If the value of the class is included in the response, then it is the actual class (see rationale in Get). If the value of the class in the request does not correspond to any of the values in the allomorphs attribute, then the agent returns an error "no such object class" to the manager.

Case g: Agent does not support allomorphism. The requested class is not recognized but an object with that name is available. The agent may either perform the operation replacing with defaults appropriate for the actual class or reject the request. If the request was confirmed and the agent rejects the request, then the agent responds with either "no such object class" or "class instance conflict" error. If it performs the operation successfully, then either an acknowledgment or a success with the modified values is returned. The returned values are appropriate for the actual class. Even though the response is not required to include the class and instance for single object request, it is recommended that the actual class and name be included in this case. This provides the manager with the information that the actual class is an extended class of the compatible class the manager understands (manager provided interoperability). If the modification is performed with partial success, then an error is sent. The use of the class field is the same as for the success case.

Case h: Requested class corresponds to the actual class in the agent or the actual class is used because the request contained the special object identifier value. Irrespective of whether allomorphism is supported or not, the agent replaces with default all the values corresponding to the attribute identifiers in the request (assuming all the attributes in the request are supported by the agent). If it performs the operation successfully, then either an acknowledgment or a success with the modified values is returned. Even though the response is not required to include the class and instance for single object request, it is recommended that the actual class and name be included in this case when the special object identifier is used in the request. If the name is not recognized, then "no such object instance" error is returned.

I.2.3.3 Summary SET operation

The table below summarizes the relationship of the manager and agent cases.

Manager case	Agent case
1	a, b, c
2	d
3	e, f, g
4	h

I.2.4 Action operation

The action operation for any specific class may include argument and responses both with mandatory and optional parameters. If there are required parameters in the argument for the actual class that are not part of the compatible class, then they must have defaults associated with them. Even though ITU-T Rec. X.720 | ISO/IEC 10165-1 permits the addition of required parameters in the action information, it is not possible to specify this using the template notation without creating a new action. However, the requirement may be specified via the behaviour. This is because only the labels of the parameters can be used to augment an action specification. This implies the original action has a field that is ANY DEFINED BY (or information object class in ITU-T Rec. X.681 | ISO/IEC 8824-2) and is augmented with parameter template label in another class (or by creating an information object). If the required fields are to be added to an action, the only approach available is to define a new action, which has a different registration. In other words, the templates do not support deriving an action from another action by adding new fields that are mandatory in a formal ASN.1 specification.

I.2.4.1 Manager role

Case 1: Manager issues an action request with a class, name and the value(s) for the parameters of the action argument (if present). The class of the object is not the actual class in the agent but a compatible class. If a response is received (only if the action was defined as confirmed), one of cases a, b, c below is valid. If interoperability is supported by the manager, then receiving a successful response in cases a and b with a different class and additional fields in the action reply will be recognized as a valid response. Otherwise, the manager will reject the response (a ROSE reject and not from CMIP).

Case 2: Manager issues an action request with a class which is either the actual class or the special object identifier which implies the use of actual class. Not all the fields included may or may not be appropriate for that class. If a response is received (only if the request was confirmed), case d below is valid.

I.2.4.2 Agent role

Case a: Agent supports allomorphism and recognizes the name of the managed object. The managed object class in the request matches one of the values in the allomorphs attribute. The agent performs the action according to its actual class using the parameters supplied in the request. Note that if there are any additional fields required for performing the action, then default values must be available as they will not be supplied by the manager. If the request is confirmed, the response from the agent depends on the following: if the action was performed successfully and if agent provides for interoperability. The response is not required to include the class and name in this case where a single object is referenced. If the operation does not succeed, an error is returned. The agent may include the actual class to inform the manager of the implemented class or the requested class (allomorphic class). If the action succeeds, the agent responds with either an acknowledgment (a confirmation that the action was performed successfully because the action definition does not include any fields for the response) or the action response with appropriate fields. The agent may choose one of the two following methods to respond. If the agent provides for interoperability, it may include only the fields appropriate for the requested class and not the additions for the actual class. In this case the value for the class field can be omitted. In the second approach, manager provided interoperability is assumed. The response may include additional fields that were not in the action for the class in the request (new parameter templates may have been included for the extension field or application of extensibility rules in ASN.1). It is recommended to include in the class field the actual class to let the manager be aware of the implemented class.

Case b: Agent does not support allomorphism but recognizes the name of the managed object and the action is valid for its actual class. The agent may not perform the requested action based on not recognizing the class in the request (different from the actual class). If the action definition indicates unconfirmed, then the manager has no knowledge of whether the action was successful or not. Depending on the action type, the effect of that action may be deduced later (for example by doing a get operation). Consider the case where the action is confirmed. Depending on whether the agent performed the action (successfully or otherwise) it may return one of the following: an error ("no such object class" or "class instance conflict"); confirmation indicating success, specific error (if any) defined for that action or a generic CMIP error. The action is performed according to the actual class. The class and name fields are not required to be in the response. If present, it is recommended to include the actual class to inform the manager of the implemented class. If both class and name are not recognized, an error response is generated (no such object instance).

Case c: Agent does not support allomorphism and does not recognize the name in the request. An error "no such object class" or "no such object instance" is returned.

Case d: Agent recognizes the class and the name irrespective of whether it supports allomorphism or not (this corresponds to case 2 where the manager requests using a class that is a compatible class and not the extended class even though the manager knows about the extended class thus supporting manager provided interoperability or the simple case where both the manager and agent know about only one class). The agent performs action according to its actual class irrespective of whether the request contained the allomorphic class or the special object identifier. The result of the action may be successful or an error. If the action was not defined as confirmed no response is generated. If successful or an error occurs in performing the action, then appropriate result or error response is issued. If the action succeeds, the response is generated according to the definition for the actual class. If the special object identifier is used, it is recommended to include the class value for the actual class even though the class and name fields are not required for the single object case.

I.2.4.3 Summary of ACTION operation

The table below summarizes the relationship of the manager and agent cases.

Manager case	Agent case
1	a, b, c
2	d

I.2.5 Delete operation

The delete operation is defined with two options: deletes contained objects and no delete allowed unless all contained objects are deleted. The following must be noted for delete operation. Irrespective of the class, it is the name that must be recognized because two instances of the same class may have different names and/or behaviour (based on the name binding used to instantiate the object).

I.2.5.1 Manager role

Case 1: Manager issues a delete request with a class and a name. The class of the object is not the actual class in the agent but a compatible class. For response one of cases a, b, c below is valid. If interoperability is supported by the manager, then receiving a successful response in cases a and b with a different class will be recognized as a valid response. Otherwise, the manager will reject the response (a ROSE reject and not from CMIP). As noted earlier, sending a reject is not useful (thus not recommended) and the manager should provide for some level of interoperability.

Case 2: Manager issues a delete request with a class which is either the actual class or the special object identifier which implies the use of actual class. The response shown in case d below is valid.

I.2.5.2 Agent role

Case a: Agent supports allomorphism and recognizes the name of the managed object. The managed object class in the request matches one of the values in the allomorphs attribute. The agent deletes the object assuming the conditions for deletion are acceptable according to the name binding used for that instance. If the deletion is not performed, an error is generated. If the deletion succeeds, the agent responds with either an acknowledgment (a confirmation that the deletion was performed successfully). The response is not required to include the class and name in this case where a single object is referenced. It is recommended to include in the class field the actual class to let the manager be aware of the implemented class (this may not be useful for the manager since the object is deleted unlike the previous operations).

Case b: Agent does not support allomorphism but recognizes the name of the managed object. The agent may not delete the object based on not recognizing the class ("no such object class" or "class instance conflict"). If the agent performs the deletion based on the object name (assuming other conditions for deletion are met), then an acknowledgment is returned. If deletion is not possible (conditions are not met), then an error is returned. In either case, it is not required to include the class and name of the object. It is recommended to include the actual class (this may not be useful for the manager since the object is deleted unlike the previous operations).

Case c: Agent does not support allomorphism and does not recognize the name in the request. An error "no such object class" or "no such object instance" is returned.

Case d: Agent recognizes the class and the name irrespective of whether it supports allomorphism or not (this corresponds to case 2 where the manager requests using a class that is a compatible class and not the extended class even though the manager knows about the extended class thus supporting manager provided interoperability or the simple case where both the manager and agent know about only one class). The agent deletes the object assuming the conditions for deletion are acceptable according to the name binding used for that instance. If the deletion is not performed an error is generated. If the deletion succeeds, the agent responds with either an acknowledgment (a confirmation that the deletion was performed successfully). The response is not required to include the class and name in this case where a single object is referenced. It is recommended to include in the class field the actual class (if the special object identifier was used; otherwise the manager and agent has the same understanding of the class value) to let the manager be aware of the implemented class (this may not be useful for the manager since the object is deleted unlike the previous operations).

I.2.5.3 Summary of DELETE operation

The table below summarizes the relationship of the manager and agent cases.

Manager case	Agent case
1	a, b, c
2	d

I.3 CMIP notification

Notifications are much simpler than the operations mentioned in the previous clause. The notification may be sent in the confirmed or unconfirmed mode. If unconfirmed, the manager may ignore what it receives because any of the following is not recognized: class, name, event type and any field of the event information. The use of RO-Reject is always possible but this is sent at ROSE level and not recognized by CMIP. However, sending a reject does not provide for TMN interoperability.

I.3.1 Manager role

Case a: The manager understands the class, name, notification type and some of the parameters in event information. The manager ignores the unknown parameters.

Case b: The manager does not provide for interoperability. It does not understand the class. The manager should ignore the notification. The manager may choose to send a reject.

Case c: The manager understands the name, notification type and some of the parameters in event information. The manager does not understand the class. The manager may ignore the notification or determine that the class is an extended class and ignores the unknown parameters (if manager provides for interoperability). Otherwise the manager may send a reject.

Case d: The manager does not recognize class, and the name (in this case not recognizing other parameters of the notification may not matter). The manager should ignore the notification.

I.3.2 Agent role

Case 1: Agent supports allomorphism. If the agent provides for interoperability, the notification may include in the class field a value from the allomorphs attribute and all the event information (irrespective of whether the parameters are applicable to the allomorphic class or not). Normally, it is expected that the agent will use the actual class because in general the agent has no knowledge which of the allomorphs should be used in the event report. If the notification is confirmed, cases a, b or c above apply.

Case 2: Agent does not support allomorphism. The agent issues the notification using the actual class and the relevant parameters for that notification. Cases a, b, c or d above are valid.

I.3.3 Summary for NOTIFICATION

The table below summarizes the relationship of the manager and agent cases.

Manager case	Agent case
a	1, 2
b	1, 2
c	1, 2
d	2

I.4 Implementation issues

I.4.1 Protocol stack related

At several places in the discussion of allomorphism, it was required that the manager be able to ignore ASN.1 syntaxes (either from attributes only in the extended class or from notifications only in the extended class). Generally, these syntaxes are not ones that have been implemented in the manager (since the manager is attempting to do allomorphic management). In order for the manager to successfully ignore these syntaxes, they must be allowed to pass through the protocol stack without disrupting the association. This is supported by the presentation layer protocols. The presentation layer normally has a requirement to abort associations if an unknown PDU is received (see 6.4.4.3/X.226 | ISO/IEC 8823-1). However, 7.5/X.711 (CMIP) states that CMIP resolves all ANY DEFINED BY OID syntaxes. To quote 7.5/X.711:

> The corresponding ASN.1 object descriptor value shall be "CMIP-PCI".
>
> This abstract syntax is defined to include all data types resolved by the ANY DEFINED BY X productions, in which X is of type OBJECT IDENTIFIER.

I.4.2 Permitted and required values

The relationship of the values of attributes between the allomorphic classes and extended classes is similar to the relationship between super classes and their subclasses. The Required Values of the allomorphic class must be a subset of the Required Values of the extended class. In turn the Required Values of the extended class must be a subset of the Permitted Values of the allomorphic class. Finally, the Permitted Values of the extended class must be a subset of the Permitted Values of the extended class. These relationships are illustrated in Figure I.1.

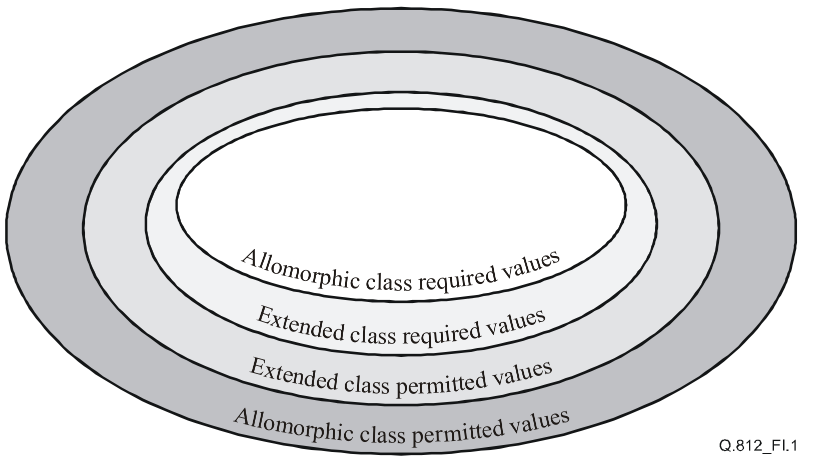

Allomorphic class required values
Extended class required values
Extended class permitted values
Allomorphic class permitted values
Q.812_FI.1

Figure I.1/Q.812 – Relationship of permitted and required values

NOTE – Even though the above rules are to be adhered to in developing the model, it may be difficult in practice. An example is the multiport circuitpack and circuitpack defined in ITU-T Rec. M.3100. The permitted values of the availability status restricted to only one value whereas practical experience showed other values are required. Hence, this restriction was removed in multiport circuitpack. However, the new version of circuitpack could not be subclassed from circuitpack as the permitted values cannot be extended and therefore strictly speaking not allomorphic. Even though the instance of an extended class has a value for an attribute outside the permitted range for the compatible class, the managing system may provide some level of interoperability. It should be able to encode the ASN.1 syntax of the attribute.

I.4.3 Initial value

A specification of a managed object class may include initial values for zero or more attributes. Unlike default values, with initial values, create request will fail if the value provided in the request is different from the specified initial value. When the value is not present, then the agent supplies the specified initial value. An extended class may specify an initial value for a given attribute different from that for the compatible class. When the agent creates the managed object, the initial value appropriate for the actual class will be used. Thus, similar to the attributes with default values discussed in I.2.1.2, the manager may receive values for attributes that are different from the ones associated with the object class in the request. It is recommended when initial values are defined for an attribute, the manager does not supply the value in the create request. This will avoid the possible rejection of the request because the supplied initial value is not appropriate for the actual class.

I.4.4 Filtering on single object

When a filtered request is made, assuming the agent has recognized the request (using conditions stated above), the following cases arise:

Case 1: The filter has all the attributes it recognizes and has been implemented for the object. The filter operation is not impacted.

Case 2: The filter has attributes that are not implemented by the agent for that object either because these are conditional or because they are for the new version of the object. The condition being checked for any attribute is equivalent to (attribute exists and the value meets the stated condition). That part of the filter should evaluate to true to make the agents simple irrespective of the implemented class. If the attribute specified in the filter is objectClass and the value to be compared is that of the compatible class, then objects with extended class will not meet the criteria. If the manager requires objects belonging to both the extended and compatible classes be selected, then the filter should include OR{equal{objectClass, x}, nonullIntersection {{x}, allomorphs}}.

I.4.5 Scoping only

The manager requests operations by providing a base object and scope level.

Case 1: The base object is the class implemented by the agent:

– Agent does not support allomorphism and responds with the actual class for the selected objects (irrespective of whether it has implemented to the extended definition it understands and implemented only to one definition). The manager may receive responses for objects with unrecognized values for the class field (because the manager does not recognize the new schema). The manager may provide for limited interoperability based on the name and other characteristics it recognizes.

– Agent supports allomorphism and has implemented to extended definitions within the selected scope. It performs the operation according to the actual classes of the objects in the scope. The response uses the actual class in the managed object class because the agent may not be aware of which one or more of the values in the allomorphs attribute the manager can recognize and the details relevant to that class (attributes, action result, etc., as noted above). Even if it supports allomorphism, it is simpler to respond in this case using the actual class and the properties appropriate for that class. The manager may provide for interoperability if it understands both versions of the definitions or limited interoperability (only one version is recognized). (If the manager supports allomorphism it may be useful for the scoped GET to request that the value of the allomorphs attribute be returned.)

Case 2: The base object is a class different from the class in the request but the name is recognized (the class is either a newer definition than what is in the request or older definition):

– Agent does not support allomorphism and implements to an older definition. It may reject the request because class is not recognized or it may use the name and respond with objects in the scope. If the manager receives "no such object class" or "class instance conflict", it may resend it with the appropriate class for the base object. This is possible only if the manager provides for interoperability and has knowledge of the version supported by the agent. It is possible the agent identifies the base object from the name (irrespective of the class), selects the objects in the scope and responds to the manager using the actual class. The manager should ignore information that it does not recognize if it provides for interoperability.

– Agent supports allomorphism and determines if the base object class is an allomorph. If this is true then it performs the operations on the selected objects (using the actual class) and responds using the actual class (manager provided interoperability).

In both the above cases, if the base object is not identified, an error is returned.

Case 3: Though unlikely, it is possible for the agent to provide for interoperability if it has the knowledge of the versions supported by the manager(s). In this case, the response may be customized to the specific manager's knowledge.

I.4.6 Scoping and filtering

When both scope and filter are in the request, anyone of the three cases discussed above is to be considered. For each case that results in selecting the objects by applying scope, the filter discussions in I.4.4, "Filtering on single object" are applied. No additional behaviour is required.

I.4.7 Naming

As noted earlier, ITU-T Rec. X.720 | ISO/IEC 10165-1 defines allomorphism as a property of the managed object. In principle, it is not required for two classes (compatible and extended) to be related by inheritance in order to exhibit allomorphic behaviour. Even though not specified in ITU-T Rec. X.720 | ISO/IEC 10165-1, it is necessary that the naming structure is the same for the two classes. Even if the managed object class parameter refers to an allomorphic class, in order for the agent to recognize the managed object, it is necessary that the managed object field uses the same structure for the actual class and the allomorphic class. The same structure implies that the sequence for constructing the local and distinguished names are the same (the superior class and RDN attributes is the same for the extended and allomorphic classes). This condition is satisfied in most cases when the two classes are related by inheritance (the name binding can include the phrase AND SUBCLASSES).

The example of the multi-port circuit pack is one where the structure rules for naming are the same as the circuit pack even though the former could not be derived as a subclass (because of the extension in permitted values). Thus, from naming perspective, it is possible to consider an instance of multi-port circuit pack to be allomorphic to circuit pack object class.

I.5 Examples of the use of allomorphism

This clause contains examples of the scenarios where the managing and managed systems implement to different versions of an information model. As a flash cut of all systems to the same version is not possible, using allomorphism to support interoperability between the systems will become important.

The figures assume that agent and manager systems are from multiple suppliers. Thus, the release numbers and the relationship to implemented version of the information model are not correlated among the suppliers.

Figure I.2 describes the simple scenario. The schema for management information model (SMK) corresponds to exactly the same definitions (both are from the interface perspective at the same release). Different numbering is used to convey the possibility that when different vendors supply the systems, they may use different release numbering options.

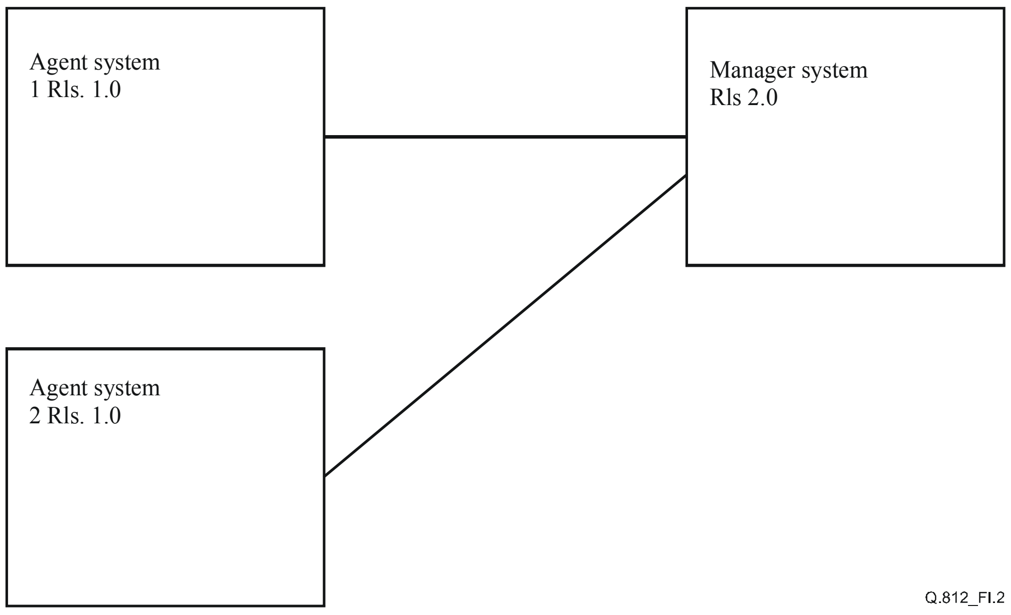


Figure I.2/Q.812 – Scenario 1

In this case, interoperability does not require the support or otherwise of allomorphism. Both agent systems and manager system are not expected to send or receive management information different from that defined by the schema.

Figure I.3 describes the following scenario: The schema for the management information model (SMK) the manager implements has more capability than Agent System Rls 1.0. The management system manages more than one agent system (different suppliers). Agent System 2 Rls 1.5 and

Manager System Rls 3.0 implements the same features (SMK is the same from interface perspective). Agent System 2 was also managed by Manager System 2 which did not upgrade to include the new features.

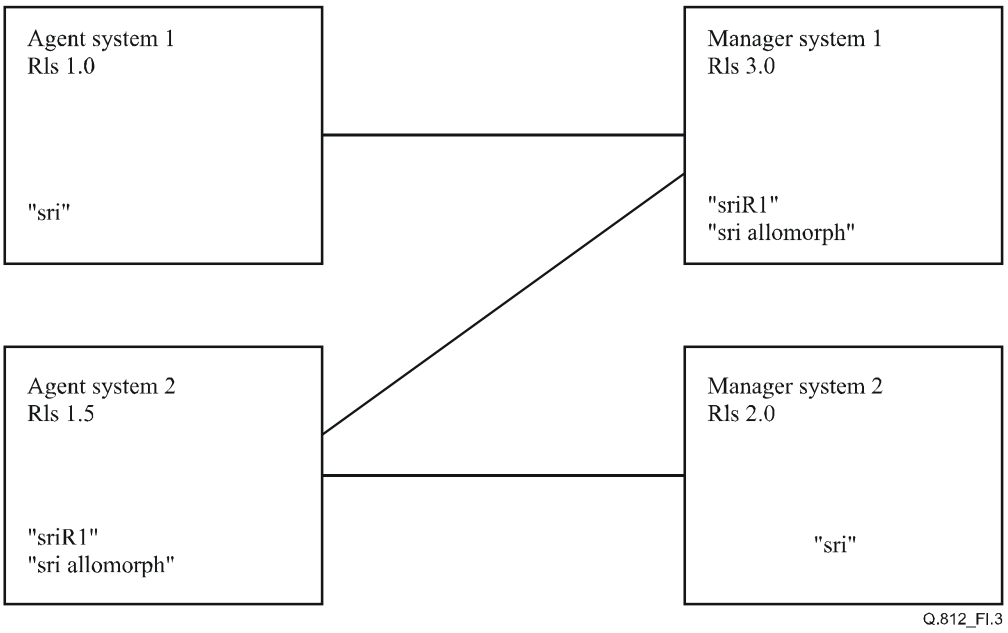


Figure I.3/Q.812 – Scenario 2

Case 1: Manager Provided Interoperability: For simplicity let us take a single managed object class "sri" and an extended class "sriR1. The interoperability solutions can be explained using this simple case without loss of generality (even though a system may choose to support allomorphism for some classes and not others). Assume that Manager System in Rls 3 can manage extensions that are not offered by Agent System 1 but by Agent System 2.

Case 2: Support or otherwise of allomorphism by Agent System 1 is not relevant. Agent System 2 supports allomorphism and has implemented the class "sriR1". The allomorph attribute contains the value "sri". Interactions between Manager System 1 and the two agents are the same as in case 1. Manager 2 does not understand the additional capabilities in "sriR1" and for notifications from Agent System 2 (using the extended class sriR1), the behaviour may not be the same as in case 1. For notifications only defined for sriR1, Manager 2 will not know how to process them and should therefore ignore them as far as management activity is concerned. Because Agent System 2 supports allomorphism, when the manager requests an operation using class "sri", it will perform the operation according to specifications for the actual class "sriR1". The agent may inform the manager that the actual class is "sriR1" by including it in the response (this is not required if the request was directed to a specific instance). Based on the requested operation, the information provided will match that of the actual class. Manager 2 should be able to ignore unrecognized information without disrupting the association. This implies that the manager should provide for some minimum level of interoperability.

Appendix II

Changes to ISP conformance requirements

Table II.1/Q.812 – Transport layer

Base standard (ITU-T Rec. X.224 | ISO/IEC 8073)				**ISP**
Ident.	**Feature**	**Clause**	**Status**	**Status**
NAC2	Class 2	6.5.4 h	NC2: None 0,1,2	NC2: at least 0
NAC4	Class 4	6.5.4 h	NC4: None 0,1,2,3,4	NC4: at least 0
NEF2	Class 2	6.5.4 k	I2R2, T2F14:O	I2R2, T2F14:oo
NEF5	Class 3	6.5.4 k	I3R2, T3F14:O	I0R2, T0F14:oo I2R2, T2F14:oo
NEF6	Class 4	6.5.4 k	I4R2, T4F14:OO	I2R2, T2F14:oo I4R2, T4F14:oo
RC4	What classes can you respond with if CR proposes only Class 4?	6.5.4 h Table 3	I4R2 or I2R2:4 or 2	I2R2:2, I4R2:4
RC4a	What classes can you respond with if CR proposes Class 4 as preferred class and the alternative class parameter is present?	6.5.4 h Table 3	I4R2:4, I2R2:2, I0R2:0 depending on coding of alternative class	I4R2:4, I2R2:2, I0R2:0 depending on coding of alternative class
S2	Support of NCMS function	Annex B	O	oi
S3	Support of Class 4 over CLNS		O	oi
TED6	Class 2	6.5.4 r	I2R2, T2F15:O	I2R2, T2F15:oo
TED8	Class 4	6.5.4 r	I4R2, T4F15:O	I0R2, T0F15:ox
NUC1	Is "Non-use of checksum" proposed in CR?	6.5.4 m	I4R1:mo	I4R1:mo
NUC2		6.5.4 m	I4R2:O	I4R2:mo

Printed in Switzerland
Geneva, 2004