INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.832.3

(01/2001)

SERIES Q: SWITCHING AND SIGNALLING

Q3 interface

Broadband access coordination

ITU-T Recommendation Q.832.3

(Formerly CCITT Recommendation)

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4 AND No. 5	Q.120–Q.249
SPECIFICATIONS OF SIGNALLING SYSTEM No. 6	Q.250–Q.309
SPECIFICATIONS OF SIGNALLING SYSTEM R1	Q.310–Q.399
SPECIFICATIONS OF SIGNALLING SYSTEM R2	Q.400–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.799
Q3 INTERFACE	**Q.800–Q.849**
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1699
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000	Q.1700–Q.1799
BROADBAND ISDN	Q.2000–Q.2999

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation Q.832.3

Broadband access coordination

Summary

This Recommendation specifies the X interface between the Operations System of a Service Node and the Operations System of an Access Network for the coordination of the management associated with VB5.1 and VB5.2 traffic interfaces and the VB5 Q3 interfaces.

Source

ITU-T Recommendation Q.832.3 was prepared by ITU-T Study Group 4 (2001-2004) and approved under the WTSA Resolution 1 procedure on 19 January 2001.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

ITU-T Recommendation Q.832.3

Broadband access coordination

1 Introduction

1.1 Purpose and scope

This Recommendation specifies the X interface between the Operations System (OS) of a Service Node (SN) and the Operations System (OS) of an Access Network (AN) for the coordination of the management associated with VB5.1 and VB5.2 traffic interfaces [2] and [3] and the VB5 Q3 interfaces [4] and [5].

Existing protocols are used where possible, and the focus of the work is on defining the object model. The definition of the functionality of TMN Operations Systems is outside the scope of this Recommendation.

Security management is also outside the scope of this Recommendation.

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; all 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 currently valid ITU-T Recommendations is regularly published.

[1] ITU-T G.902 (1995), Framework Recommendation on functional access networks (AN) – Architecture and functions, access types, management and service node aspects.

[2] ITU-T G.967.1 (1998), *V-interfaces at the service node (SN) – VB5.1 reference point specification*.

[3] ITU-T G.967.2 (1999), *V-interfaces at the service node (SN) – VB5.2 reference point specification*.

[4] ITU-T Q.832.1 (1998), *VB5.1 Management.*

[5] ITU-T Q.832.2 (1998), *VB5.2 Management.*

[6] ITU-T X.721 (1992) | ISO/IEC 10165-2:1992, *Information technology – Open Systems Interconnection – Structure of management information: Definition of management information*.

[7] IETF RFC 1057(1988): *RPC: Remote Procedure Call Protocol Specification: Version 2.*

3 Definitions, abbreviations, and conventions

3.1 Definitions

This Recommendation defines the following terms:

3.1.1 VB5 Resources: The management of user port functions and service port functions providing user network interface (UNI) and service node interface (SNI) functionality, respectively, are considered in this Recommendation based on the framework defined in ITU-T G.902 [1]. Transmission specific resources lie outside its scope. VB5 Resources are referred to in this Recommendation as resources.

In addition this Recommendation uses terms defined in other ITU-T Recommendations as follows:

ITU-T G.902 [1]: access network (AN), user port functions, service node (SN), service node interface (SNI), service port functions.

ITU-T G.967.1 [2]: logical service port (LSP), logical user port (LUP), physical service port (PSP), physical user port (PUP), real time management coordination (RTMC), virtual user port (VUP).

ITU-T G.967.2 [3]: Broadband bearer connection control (B-BCC)

3.2 Abbreviations

This Recommendation uses the following abbreviations:

AN Access Network

ASN.1 Abstract Syntax Notation one

ATM Asynchronous Transfer Mode

B-BCC Broadband Bearer Connection Control

IETF Internet Engineering Task Force

LSP Logical Service Port

LUP Logical User Port

OS Operations System

RFC Request For Comment

RTMC Real Time Management Coordination

SN Service Node

SNI Service Node Interface

TMN Telecommunication Management Network

TTP Trail Termination Point

UNI User-Network Interface

VC Virtual Channel

VP Virtual Path

VPC Virtual Path Connection

VPCI Virtual Path Connection Identifier

3.3 Conventions

Objects and their characteristics and associated ASN.1 defined here are given names with capitals used to indicate the start of the next word and acronyms are treated as if they were words.

Throughout this Recommendation, all new attributes are named according to the following guidelines:

– The name of an attribute ends in the string "Ptr" if and only if the attribute value is intended to identify a single object.

– The name of an attribute ends in the string "PtrList" if and only if the attribute value is intended to identify one or more objects.

– The name of an attribute is composed of the name of an object class followed by the string "Ptr" if and only if the attribute value is intended to identify a specific object class.

– If an attribute is intended to identify different object classes, a descriptive name is given to that attribute and a description is provided in the attribute behaviour.

– The name of an attribute ends in the string "Id" if and only if the attribute value is intended to identify the name of an object, in which case this attribute should be the first one listed, should use ASN.1 NameType and should not be used to convey other information.

– The name of an attribute is composed of the name of an object class followed by the string "Id" if and only if the attribute value is intended to identify the name of the object class holding that attribute.

4 General overview

The following information model diagrams have been drawn for the purpose of clarifying the relations between the different object classes of the model.

1) Entity Relationship Models showing the relations of the different managed objects.

2) Inheritance Hierarchy showing how managed objects are derived from each other (i.e. the different paths of inherited characteristics of the different managed objects).

These diagrams are only for clarification. The formal specification in terms of GDMO templates and ASN.1 type definitions are the relevant information for implementations.

4.1 Entity-Relationship Models

The following conventions are used in the diagrams:

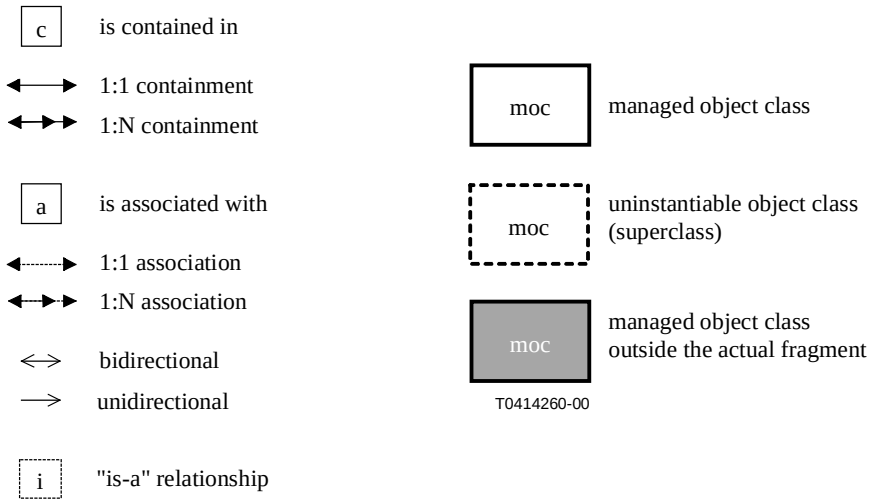

Figure 1/Q.832.3 – Conventions used in diagrams for Entity Relationship Models

Where the directionality of containment is not clear it can be identified by implications since the root class is unique.

4.1.1 Entity relationship diagram

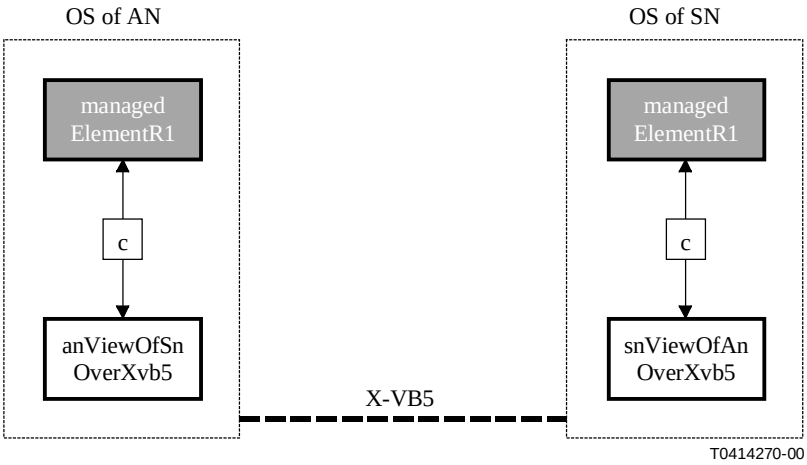

Figure 2/Q.832.3 – Entity relationship diagram

4.2 Inheritance hierarchy

The Figure 3 traces the inheritance relationships from the highest level object (ITU-T X.721, "top") to the managed objects which are defined in this Recommendation.

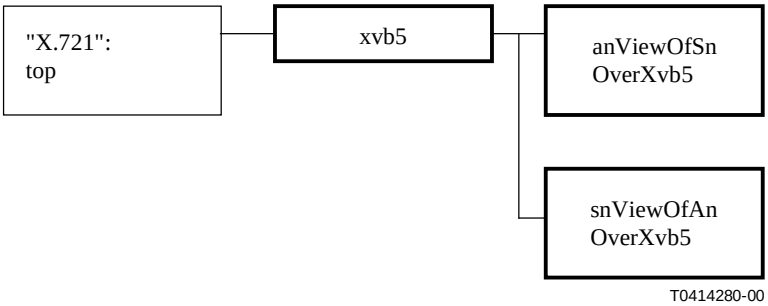

Figure 3/Q.832.3 – Inheritance Hierarchy

5 Formal definitions

This clause gives the formal definitions of the managed object classes, name bindings, general packages, behaviours, attributes, actions and notifications.

5.1 Object classes

This clause specifies the object classes for all of the managed objects used in the management information model. These object classes are either defined here or by reference to other specifications. Classes of managed objects which are defined elsewhere and which are only used for containment are not included, but are identified by the name bindings for the classes specified here.

5.1.1 Profiling notes for imported classes

There are no imported classes.

5.1.2 Definition of classes

5.1.2.1 anViewOfSnOverXvb5 (AN view of SN over X-VB5)

```
anViewOfSnOverXvb5 MANAGED OBJECT CLASS
    DERIVED FROM xvb5;
    CHARACTERIZED BY
        anViewOfSnOverXvb5Pkg PACKAGE
            BEHAVIOUR anViewOfSnOverXvb5Beh;
            ACTIONS
                anServiceLabelInquiry;;;
REGISTERED AS {q832-3ManagedObjectClass 1};

anViewOfSnOverXvb5Beh BEHAVIOUR
    DEFINED AS
        "This managed object represents the Service Node side of an X-VB5 interface, as seen by the Operations
        System of the Access Network.";
```

5.1.2.2 snViewOfAnOverXvb5 (SN view of AN over X-VB5)

```
snViewOfAnOverXvb5 MANAGED OBJECT CLASS
    DERIVED FROM xvb5;
    CHARACTERIZED BY
        snViewOfAnOverXvb5Pkg PACKAGE
            BEHAVIOUR snViewOfAnOverXvb5Beh;
            ACTIONS
                addAnLoopRequest,
                snAccessLabelsInquiry,
                removeAnLoopRequest;;;
REGISTERED AS {q832-3ManagedObjectClass 2};

snViewOfAnOverXvb5Beh BEHAVIOUR
    DEFINED AS
        "This managed object represents the Access Network side of an X-VB5 interface, as seen by the Operations
        System of the Service Node.";
```

5.1.2.3 xvb5 (X-VB5)

```
xvb5 MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X.721|ISO/IEC 10165-2":top;
    CHARACTERIZED BY
        "ITU-T M.3100":operationalStatePackage,
        xvb5Pkg PACKAGE
            BEHAVIOUR xvb5Beh;
            ATTRIBUTES
                xvb5Id
                    GET SET-BY-CREATE,
                " Rec. X.721 | ISO/IEC 10165-2": administrativeState
                    GET-REPLACE;
            ACTIONS
                addLupsRequest,
                addVb5ConnectionRequest,
                addVb5InterfaceRequest,
                addVb5ProtocolRequest,
                addVb5ProtocolVpRequest,
                addVb5VcsRequest,
                addVb5VpsRequest,
                auditVb5ConnectionRequest,
                auditVb5VpciRequest,
                listLupsRequest,
```

```
            listVb5ProtocolDetailsRequest,
            listVb5InterfacesRequest,
            listVb5VcsRequest,
            listVb5VpsRequest,
            removeLupsRequest,
            removeVb5ConnectionRequest,
            removeVb5InterfaceRequest,
            removeVb5ProtocolRequest,
            removeVb5ProtocolVpRequest,
            removeVb5VcsRequest,
            removeVb5VpsRequest;
        NOTIFICATIONS
            addLupsIndication,
            addVb5ConnectionIndication,
            addVb5InterfaceIndication,
            addVb5ProtocolIndication,
            addVb5ProtocolVpIndication,
            addVb5VcsIndication,
            addVb5VpsIndication,
            removeLupsIndication,
            removeVb5ConnectionIndication,
            removeVb5InterfaceIndication,
            removeVb5ProtocolIndication,
            removeVb5ProtocolVpIndication,
            removeVb5VcsIndication,
            removeVb5VpsIndication,
            resourceStatusIndication,
            " Rec. X.721 | ISO/IEC 10165-2": stateChange,
            " Rec. X.721 | ISO/IEC 10165-2": objectCreation,
            " Rec. X.721 | ISO/IEC 10165-2": objectDeletion;;;
REGISTERED AS {q832-3ManagedObjectClass 3};

xvb5Beh BEHAVIOUR
    DEFINED AS
        "The xvb5 managed object class represents the aspects of an X-VB5 interface that are common to both sides.
        The xvb5 class is not instantiated.";
```

5.2 Name bindings

5.2.1 xvb5-managedElementR1

```
xvb5-managedElementR1 NAME BINDING
    SUBORDINATE OBJECT CLASS xvb5 AND SUBCLASSES;
    NAMED BY SUPERIOR OBJECT CLASS "ITU-T Rec. M.3100":managedElementR1 AND SUBCLASSES;
    WITH ATTRIBUTE xvb5Id;
    CREATE
        WITH-REFERENCE-OBJECT,
        WITH-AUTOMATIC-INSTANCE-NAMING;
    DELETE
        DELETES-CONTAINED-OBJECTS;
REGISTERED AS {q832-3NameBinding 1};
```

5.3 Definition of packages

No packages are defined.

5.4 Definition of attributes

5.4.1 xvb5Id (X-VB5 identifier)

```
xvb5Id ATTRIBUTE
    WITH ATTRIBUTE SYNTAX Q832-3ASN1Module.NameType;
    MATCHES FOR EQUALITY;
    BEHAVIOUR xvb5IdBeh;
REGISTERED AS {q832-3Attribute 1};

xvb5IdBeh BEHAVIOUR
    DEFINED AS
        "This attribute is used for naming instances of the managed object class xvb5 and subclasses.";
```

5.5 Definition of actions

5.5.1 addAnLoopRequest (add AN loop request)

```
addAnLoopRequest ACTION
    BEHAVIOUR addAnLoopRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddAnLoopRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.AddAnLoopRequestResult;
REGISTERED AS {q832-3Action 1};

addAnLoopRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used by the OS of the SN to request the OS of the AN to loop a connection so that cells sent to
        the AN will be returned.";
```

5.5.2 addLupsRequest (add LUPs request)

```
addLupsRequest ACTION
    BEHAVIOUR addLupsRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddLupsRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.AddLupsRequestResult;
REGISTERED AS {q832-3Action 2};

addLupsRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to add Logical User Ports to a VB5 interface.";
```

5.5.3 addVb5ConnectionRequest (add VB5 connection request)

```
addVb5ConnectionRequest ACTION
    BEHAVIOUR addVb5ConnectionRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5ConnectionRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5ConnectionRequestResult;
REGISTERED AS {q832-3Action 3};

addVb5ConnectionRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to add a connection associated with a VB5 interface. The egress
        direction is out of the Access Network towards Service Node. The ingress direction is into the Access Network
        from the Service Node.";
```

5.5.4 addVb5InterfaceRequest (add VB5 interface request)

```
addVb5InterfaceRequest ACTION
    BEHAVIOUR addVb5InterfaceRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5InterfaceRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5InterfaceRequestResult;
REGISTERED AS {q832-3Action 4};

addVb5InterfaceRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to add a VB5 interface.";
```

5.5.5 addVb5ProtocolRequest (add VB5 protocol request)

```
addVb5ProtocolRequest ACTION
    BEHAVIOUR addVb5ProtocolRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5ProtocolRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5ProtocolRequestResult;
REGISTERED AS {q832-3Action 5};

addVb5ProtocolRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to add a protocol to an existing VB5 interface.";
```

5.5.6 addVb5ProtocolVpRequest (add VB5 protocol VP request)

```
addVb5ProtocolVpRequest ACTION
    BEHAVIOUR addVb5ProtocolVpRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5ProtocolVpRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5ProtocolVpRequestResult;
REGISTERED AS {q832-3Action 6};

addVb5ProtocolVpRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to add a protocol VP to a VB5 interface.";
```

5.5.7 addVb5VcsRequest (add VB5 VCs request)

```
addVb5VcsRequest ACTION
    BEHAVIOUR addVb5VcsRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5VcsRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5VcsRequestResult;
REGISTERED AS {q832-3Action 7};

addVb5VcsRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to add VCs to a VP which is associated with a VB5 interface.";
```

5.5.8 addVb5VpsRequest (add VB5 VPs request)

```
addVb5VpsRequest ACTION
    BEHAVIOUR addVb5VpsRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5VpsRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5VpsRequestResult;
REGISTERED AS {q832-3Action 8};
```

```
addVb5VpsRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to add VPs that are associated with a VB5 interface.";
```

5.5.9 anServiceLabelInquiry (AN service label inquiry)

```
anServiceLabelInquiry ACTION
    BEHAVIOUR anServiceLabelInquiryBeh;
    MODE CONFIRMED;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.AnServiceLabelInquiryResult;
REGISTERED AS {q832-3Action 9};

anServiceLabelInquiryBeh BEHAVIOUR
    DEFINED AS
        "This action is used by the OS of an AN to inquire the label that an SN uses for the AN.";
```

5.5.10 auditVb5ConnectionRequest (audit VB5 connection request)

```
auditVb5ConnectionRequest ACTION
    BEHAVIOUR auditVb5ConnectionRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AuditVb5ConnectionRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.AuditVb5ConnectionRequestResult;
REGISTERED AS {q832-3Action 10};

auditVb5ConnectionRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to audit a connection which is associated with a VB5 interface.";
```

5.5.11 auditVb5VpciRequest (audit VB5 VPCI request)

```
auditVb5VpciRequest ACTION
    BEHAVIOUR auditVb5VpciRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AuditVb5VpciRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.AuditVb5VpciRequestResult;
REGISTERED AS {q832-3Action 11};

auditVb5VpciRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to audit a VPCI which is associated with a VB5 interface.";
```

5.5.12 listLupsRequest (list LUPs request)

```
listLupsRequest ACTION
    BEHAVIOUR listLupsRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.ListLupsRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.ListLupsRequestResult;
REGISTERED AS {q832-3Action 12};

listLupsRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to list the Logical User Ports associated with a VB5 interface
        between an Access Network and a Service Node which the two Operations Systems together control.";
```

5.5.13 listVb5ProtocolDetailsRequest (list protocol details request)

```
listVb5ProtocolDetailsRequest ACTION
    BEHAVIOUR listVb5ProtocolDetailsRequestBeh;
    MODE CONFIRMED;
```

```
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.ListVb5ProtocolDetailsRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.ListVb5ProtocolDetailsRequestResult;
REGISTERED AS {q832-3Action 13};

listVb5ProtocolDetailsRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to list the details of the protocols of a VB5 interface between an
        Access Network and a Service Node which the two Operations Systems together control.";
```

5.5.14 listVb5InterfacesRequest (list VB5 interfaces request)

```
listVb5InterfacesRequest ACTION
    BEHAVIOUR listVb5InterfacesRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.ListVb5InterfacesRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.ListVb5InterfacesRequestResult;
REGISTERED AS {q832-3Action 14};

listVb5InterfacesRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to list the identities of the VB5 interfaces between Access
        Network(s) and the Service Node(s) which the two Operations Systems together control.";
```

5.5.15 listVb5VcsRequest (list VB5 VCs request)

```
listVb5VcsRequest ACTION
    BEHAVIOUR listVb5VcsRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.ListVb5VcsRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.ListVb5VcsRequestResult;
REGISTERED AS {q832-3Action 15};

listVb5VcsRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to list the VCs associated with a VB5 interface.";
```

5.5.16 listVb5VpsRequest (list VB5 VPs request)

```
listVb5VpsRequest ACTION
    BEHAVIOUR listVb5VpsRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.ListVb5VpsRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.ListVb5VpsRequestResult;
REGISTERED AS {q832-3Action 16};

listVb5VpsRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to list the VPs associated with a VB5 interface.";
```

5.5.17 removeAnLoopRequest (remove AN loop request)

```
removeAnLoopRequest ACTION
    BEHAVIOUR removeAnLoopRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveAnLoopRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.RemoveAnLoopRequestResult;
REGISTERED AS {q832-3Action 17};

removeAnLoopRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used by the OS of the SN to request the OS of the AN to remove a loop from a connection.";
```

5.5.18 removeLupsRequest (remove LUPs request)

```
removeLupsRequest ACTION
    BEHAVIOUR removeLupsRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveLupsRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.RemoveLupsRequestResult;
REGISTERED AS {q832-3Action 18};

removeLupsRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to remove Logical User Ports from a VB5 interface.";
```

5.5.19 removeVb5ConnectionRequest (remove VB5 connection request)

```
removeVb5ConnectionRequest ACTION
    BEHAVIOUR removeVb5ConnectionRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5ConnectionRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5ConnectionRequestResult;
REGISTERED AS {q832-3Action 19};

removeVb5ConnectionRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to remove a connection associated with a VB5 interface. The egress
        direction is out of the Access Network towards Service Node. The ingress direction is into the Access Network
        from the Service Node.";
```

5.5.20 removeVb5InterfaceRequest (remove VB5 interface request)

```
removeVb5InterfaceRequest ACTION
    BEHAVIOUR removeVb5InterfaceRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5InterfaceRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5InterfaceRequestResult;
REGISTERED AS {q832-3Action 20};

removeVb5InterfaceRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to remove a VB5 interface.";
```

5.5.21 removeVb5ProtocolRequest (remove VB5 protocol request)

```
removeVb5ProtocolRequest ACTION
    BEHAVIOUR removeVb5ProtocolRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5ProtocolRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5ProtocolRequestResult;
REGISTERED AS {q832-3Action 21};

removeVb5ProtocolRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to remove a protocol from a VB5 interface.";
```

5.5.22 removeVb5ProtocolVpRequest (remove VB5 protocol Vp request)

```
removeVb5ProtocolVpRequest ACTION
    BEHAVIOUR removeVb5ProtocolVpRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5ProtocolVpRequestInfo;
```

```
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5ProtocolVpRequestResult;
REGISTERED AS {q832-3Action 22};

removeVb5ProtocolVpRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to remove the protocol VP from a VB5 interface.";
```

5.5.23 removeVb5VcsRequest (remove VB5 VCs request)

```
removeVb5VcsRequest ACTION
    BEHAVIOUR removeVb5VcsRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5VcsRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5VcsRequestResult;
REGISTERED AS {q832-3Action 23};

removeVb5VcsRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to remove VCs from a VP which is associated with a VB5
        interface.";
```

5.5.24 removeVb5VpsRequest (remove VB5 VPs request)

```
removeVb5VpsRequest ACTION
    BEHAVIOUR removeVb5VpsRequestBeh;
    MODE CONFIRMED;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5VpsRequestInfo;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5VpsRequestResult;
REGISTERED AS {q832-3Action 24};

removeVb5VpsRequestBeh BEHAVIOUR
    DEFINED AS
        "This action is used to request the peer OS to remove VPs that are associated with a VB5 interface.";
```

5.5.25 snAccessLabelsInquiry (SN access labels inquiry)

```
snAccessLabelsInquiry ACTION
    BEHAVIOUR snAccessLabelsInquiryBeh;
    MODE CONFIRMED;
    WITH REPLY SYNTAX Q832-3ASN1DefinedTypesModule.SnAccessLabelsInquiryResult;
REGISTERED AS {q832-3Action 25};

snAccessLabelsInquiryBeh BEHAVIOUR
    DEFINED AS
        "This action is used by the OS of an SN to inquire the access labels that an AN uses for the SN and the VB5
        interface.";
```

5.6 Definition of notifications

5.6.1 addLupsIndication (add LUPs indication)

```
addLupsIndication NOTIFICATION
    BEHAVIOUR addLupsIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddLupsIndicationInfo;
REGISTERED AS {q832-3Notification 1};

addLupsIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the addition of Logical User Ports to a VB5 interface.";
```

5.6.2 addVb5ConnectionIndication (add VB5Connection indication)

addVb5ConnectionIndication NOTIFICATION
    BEHAVIOUR addVb5ConnectionIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5ConnectionIndicationInfo;
REGISTERED AS {q832-3Notification 2};

addVb5ConnectionIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the addition of a connection associated with a VB5 interface. The egress direction is out of the Access Network towards Service Node. The ingress direction is into the Access Network from the Service Node.";

5.6.3 addVb5InterfaceIndication (add VB5 interface indication)

addVb5InterfaceIndication NOTIFICATION
    BEHAVIOUR addVb5InterfaceIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5InterfaceIndicationInfo;
REGISTERED AS {q832-3Notification 3};

addVb5InterfaceIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to inform the peer OS that a new VB5 interface has been added.";

5.6.4 addVb5ProtocolIndication (add VB5 protocol indication)

addVb5ProtocolIndication NOTIFICATION
    BEHAVIOUR addVb5ProtocolIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5ProtocolIndicationInfo;
REGISTERED AS {q832-3Notification 4};

addVb5ProtocolIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the addition of a protocol to an existing VB5 interface.";

5.6.5 addVb5ProtocolVpIndication (add VB5 protocol VP indication)

addVb5ProtocolVpIndication NOTIFICATION
    BEHAVIOUR addVb5ProtocolVpIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5ProtocolVpIndicationInfo;
REGISTERED AS {q832-3Notification 5};

addVb5ProtocolVpIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the addition of a protocol VP to a VB5 interface.";

5.6.6 addVb5VcsIndication (add VB5 VCs indication)

addVb5VcsIndication NOTIFICATION
    BEHAVIOUR addVb5VcsIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5VcsIndicationInfo;
REGISTERED AS {q832-3Notification 6};

addVb5VcsIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the addition of VCs to a VP which is associated with a VB5 interface.";

5.6.7 addVb5VpsIndication (add VB5 VPs indication)

addVb5VpsIndication NOTIFICATION
    BEHAVIOUR addVb5VpsIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.AddVb5VpsIndicationInfo;
REGISTERED AS {q832-3Notification 7};

addVb5VpsIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the addition of VPs that are associated with a VB5 interface.";

5.6.8 removeLupsIndication (remove LUPs indication)

removeLupsIndication NOTIFICATION
    BEHAVIOUR removeLupsIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveLupsIndicationInfo;
REGISTERED AS {q832-3Notification 8};

removeLupsIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the removal of Logical User Ports from a VB5 interface.";

5.6.9 removeVb5ConnectionIndication (remove VB5Connection indication)

removeVb5ConnectionIndication NOTIFICATION
    BEHAVIOUR removeVb5ConnectionIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5ConnectionIndicationInfo;
REGISTERED AS {q832-3Notification 9};

removeVb5ConnectionIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the removal of a connection associated with a VB5 interface. The egress direction is out of the Access Network towards Service Node. The ingress direction is into the Access Network from the Service Node.";

5.6.10 removeVb5InterfaceIndication (remove VB5 interface indication)

removeVb5InterfaceIndication NOTIFICATION
    BEHAVIOUR removeVb5InterfaceIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5InterfaceIndicationInfo;
REGISTERED AS {q832-3Notification 10};

removeVb5InterfaceIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the removal of a VB5 interface.";

5.6.11 removeVb5ProtocolIndication (remove VB5 protocol indication)

removeVb5ProtocolIndication NOTIFICATION
    BEHAVIOUR removeVb5ProtocolIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5ProtocolIndicationInfo;
REGISTERED AS {q832-3Notification 11};

removeVb5ProtocolIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the removal of a protocol from a VB5 interface.";

5.6.12 removeVb5ProtocolVpIndication (remove VB5 protocol VP indication)

removeVb5ProtocolVpIndication NOTIFICATION
    BEHAVIOUR removeVb5ProtocolVpIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5ProtocolVpIndicationInfo;
REGISTERED AS {q832-3Notification 12};

removeVb5ProtocolVpIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the removal of the protocol VP from a VB5 interface.";

5.6.13 removeVb5VcsIndication (remove VB5 VCs indication)

removeVb5VcsIndication NOTIFICATION
    BEHAVIOUR removeVb5VcsIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5VcsIndicationInfo;
REGISTERED AS {q832-3Notification 13};

removeVb5VcsIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the removal of VCs from a VP which is associated with a VB5 interface.";

5.6.14 removeVb5VpsIndication (remove VB5 VPs indication)

removeVb5VpsIndication NOTIFICATION
    BEHAVIOUR removeVb5VpsIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.RemoveVb5VpsIndicationInfo;
REGISTERED AS {q832-3Notification 14};

removeVb5VpsIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the removal of VPs that are associated with a VB5 interface.";

5.6.15 resourceStatusIndication (resource status indication)

resourceStatusIndication NOTIFICATION
    BEHAVIOUR resourceStatusIndicationBeh;
    WITH INFORMATION SYNTAX Q832-3ASN1DefinedTypesModule.ResourceStatusIndicationInfo;
REGISTERED AS {q832-3Notification 15};

resourceStatusIndicationBeh BEHAVIOUR
    DEFINED AS
        "This notification is used to notify the peer OS of the change of status of a Resource.";

6 Type definitions

Q832-3ASN1DefinedTypesModule {
itu-t (0) recommendation (0) q (17) q832 (832) dot (127) coord (3)
informationModel (0) asn1Modules (2) asn1DefinedTypesModule (0)}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN
-- EXPORTS everything

IMPORTS
ObjectInstance
FROM CMIP-1 {joint-iso-itu-t ms (9) cmip (1) modules (0) protocol (3)}

```
        NameType
        FROM ASN1DefinedTypesModule {ccitt recommendation m 3100
                informationModel(0) asn1Modules(2) asn1DefinedTypesModule(0)}
        VciValue, VpiValue
        FROM AtmMIBMod {itu-t (0) recommendation (0) i (9) atmm (751)
                informationModel (0) asn1Module (2) atm(0)}
        VpciValue
        FROM AtmMIBMod {itu-t (0) recommendation (0) q (17) 824 (824) dot (127) bsm (6)
                informationModel (0) asn1Modules (2) asn1DefinedTypesModule (0)}
;  -- end of imports

-- start of object identifier definitions

q832-3InformationModel
        OBJECT IDENTIFIER ::=
        {itu-t (0) recommendation (0) q (17) q832 (832) dot (127) coord(3) informationModel(0)}
q832-3StandardSpecificExtension
        OBJECT IDENTIFIER ::= {informationModel q832-3StandardSpecificExtension(0)}
q832-3ManagedObjectClass
        OBJECT IDENTIFIER ::= {informationModel q832-3ManagedObjectClass(3)}
q832-3Package
        OBJECT IDENTIFIER ::= {informationModel q832-3Package(4)}
q832-3NameBinding
        OBJECT IDENTIFIER ::= {informationModel q832-3NameBinding(6)}
q832-3Attribute
        OBJECT IDENTIFIER ::= {informationModel q832-3Attribute (7)}
q832-3Action
        OBJECT IDENTIFIER ::= {informationModel q832-3Action(9)}
q832-3Notification
        OBJECT IDENTIFIER ::= {informationModel q832-3Notification(10)}

-- end of object identifier definitions

-- other ASN.1 definitions in alphabetical order – all these are new

AddAnLoopRequestInfo ::= SEQUENCE {
        logicalServicePortNumber[0]     INTEGER,
        logicalUserPortNumber   [1]     INTEGER OPTIONAL,
        vpciValue               [2]     VpciValue,
        vciValue                [3]     VciValue OPTIONAL}

AddAnLoopRequestResult ::= CHOICE {
        loopAdded       [0]     NULL,
        loopNotAdded    [1]     LoopNotAddedInfo}

AddLupsIndicationInfo ::= AddLupsRequestInfo

AddLupsRequestInfo ::= SEQUENCE {
        logicalServicePortNumber[0]     INTEGER,
        logicalUserPortNumber   [1]     SEQUENCE OF INTEGER}

AddLupsRequestResult ::= INTEGER {
        lupAdded        (0),
        lupNotAdded     (1)}

AddVb5ConnectionIndicationInfo ::= AddVb5ConnectionRequestInfo

AddVb5ConnectionRequestInfo ::= SEQUENCE {
        egressPeakCellRateCLP0Plus1             [0]     INTEGER,
        egressPeakCellRateCLP0                  [1]     INTEGER,
        ingressPeakCellRateCLP0Plus1            [2]     INTEGER,
        ingressPeakCellRateCLP0                 [3]     INTEGER,
        egressSustainableCellRateCLP0Plus1      [4]     INTEGER,
```

```
        egressSustainableCellRateCLP0          [5]   INTEGER,
        ingressSustainableCellRateCLP0Plus1    [6]   INTEGER,
        ingressSustainableCellRateCLP0         [7]   INTEGER,
        egressCDVToleranceCLP0Plus1            [8]   INTEGER,
        egressCDVToleranceCLP0                 [9]   INTEGER,
        ingressCDVToleranceCLP0Plus1           [10]  INTEGER,
        ingressCDVToleranceCLP0                [11]  INTEGER,
        egressMaxBurstSizeCLP0Plus1            [12]  INTEGER,
        egressMaxBurstSizeCLP0                 [13]  INTEGER,
        ingressMaxBurstSizeCLP0Plus1           [14]  INTEGER,
        ingressMaxBurstSizeCLP0                [15]  INTEGER,
        egressQosClass                         [16]  INTEGER,
        ingressQosClass                        [17]  INTEGER,
        vciValueA                              [18]  INTEGER OPTIONAL,
        vciValueB                              [19]  INTEGER OPTIONAL,
        physicalPortA                          [20]  INTEGER,
        vpiValueA                              [21]  INTEGER,
        logicalServicePortA                    [22]  INTEGER OPTIONAL,
        vpciValueA                             [23]  INTEGER OPTIONAL,
        physicalPortB                          [24]  INTEGER,
        vpiValueB                              [25]  INTEGER,
        logicalServicePortB                    [26]  INTEGER OPTIONAL,
        vpciValueB                             [27]  INTEGER OPTIONAL}

AddVb5ConnectionRequestResult ::= INTEGER {
        vb5ConnectionAdded      (0),
        vb5ConnectionNotAdded   (1)}

AddVb5InterfaceIndicationInfo ::= AddVb5InterfaceRequestInfo

AddVb5InterfaceRequestInfo ::= INTEGER   -- Logical service port number

AddVb5InterfaceRequestResult ::= INTEGER {
        vb5InterfaceAdded       (0),
        vb5InterfaceNotAdded    (1)}

AddVb5ProtocolIndicationInfo ::= AddVb5ProtocolRequestInfo

AddVb5ProtocolRequestInfo ::= SEQUENCE {
        logicalServicePortNumber        [0]   INTEGER,
        vpciValue                       [1]   VpciValue,
        maxNumVciBitsNearEnd            [2]   INTEGER OPTIONAL,
        maxNumVciBitsSupported          [3]   INTEGER OPTIONAL,
        maxNumActiveVccsAllowed         [4]   INTEGER OPTIONAL,
        maxNumActiveVccsNearEnd         [5]   INTEGER OPTIONAL}

AddVb5ProtocolRequestResult ::= INTEGER {
        vb5ProtocolAdded        (0),
        vb5ProtocolNotAdded     (1)}

AddVb5ProtocolVpIndicationInfo ::= AddVb5ProtocolVpRequestInfo

AddVb5ProtocolVpRequestInfo ::= SEQUENCE {
        logicalServicePortNumber[0]   INTEGER,
        vpciValue                     [1]   VpciValue,
        vpProfile                     [2]   VpProfile OPTIONAL}

AddVb5ProtocolVpRequestResult ::= INTEGER {
        vb5ProtocolVpAdded      (0),
        vb5ProtocolVpNotAdded   (1)}

AddVb5VcsIndicationInfo ::= AddVb5VcsRequestInfo
```

```
AddVb5VcsRequestInfo ::= SEQUENCE {
        logicalServicePortNumber[0]     INTEGER,
        logicalServiceSubport    [1]    INTEGER OPTIONAL,
        vpciValue                [2]    VpciValue,
        vciValue                 [3]    VciValue}

AddVb5VcsRequestResult ::= INTEGER {
        vb5VcsAdded              (0),
        vb5VcsNotAdded           (1)}

AddVb5VpsIndicationInfo ::= AddVb5VpsRequestInfo

AddVb5VpsRequestInfo ::= SEQUENCE {
        logicalServicePortNumber[0]     INTEGER,
        logicalServiceSubport    [1]    INTEGER OPTIONAL,
        addVpInfo                [2]    AddVpInfo}

AddVb5VpsRequestResult ::= INTEGER {
        vb5VpsAdded              (0),
        vb5VpsNotAdded           (1)}

AddVpInfo ::= SEQUENCE OF SEQUENCE {
        physicalPort             [0]    INTEGER,
        vpiValue                 [1]    VpiValue,
        vpciValue                [2]    VpciValue,
vpProfile                        [3]    VpProfile OPTIONAL}

AnServiceLabelInquiryResult ::= INTEGER OPTIONAL

AuditVb5ConnectionRequestInfo ::= SEQUENCE {
        logicalServicePortNumber[0]     INTEGER,
        specifiedVpOrVc          [1]    SpecifiedVpOrVc}

AuditVb5ConnectionRequestResult ::= CHOICE {
        connectionAudited        [0]    SpecifiedVpOrVc,
        connectionNotAudited     [1]    ConnectionNotAuditedInfo}

AuditVb5VpciRequestInfo ::= SEQUENCE {
        logicalServicePortNumber            [0]    INTEGER,
        specifiedNniVpciOrRemoteVp          [1]    SpecifiedNniVpciOrRemoteVp}

AuditVb5VpciRequestResult ::= CHOICE {
        vpciAudited      [0]    SpecifiedNniVpciOrRemoteVp,
        vpciNotAudited   [1]    VpciNotAuditedInfo}

ConnectionNotAuditedInfo ::= INTEGER {
        unspecified              (0),
        unknownPhysicalPort      (1),
        unknownVpiValue          (2),
        unknownVciValue          (3)}

ListLupsRequestInfo ::= INTEGER     -- Logical service port number

ListLupsFailureInfo ::= INTEGER {
        unspecified              (0),
        unknownLspNumber         (1)}

ListLupsRequestResult ::= CHOICE {
        success          [0]    INTEGER,  -- Logical user port number
        failure          [1]    ListLupsFailureInfo}

ListVb5ProtocolDetailsRequestInfo ::= INTEGER    -- Logical service port number
```

```
ListVb5ProtocolDetailsRequestResult ::= CHOICE {
        success         [0]     ListVb5ProtocolDetailsSuccessInfo,
        failure         [1]     ListVb5ProtocolDetailsFailureInfo}

ListVb5ProtocolDetailsFailureInfo ::= INTEGER {
        unspecified                     (0),
        unknownLspNumber                (1)}

ListVb5ProtocolDetailsSuccessInfo ::= SEQUENCE OF Vb5ProtocolDetails

Vb5ProtocolDetails ::= SEQUENCE {
        protocolType                                    [0]     ProtocolType,
        vpciValue                                       [1]     VpciValue,
        vciValue                                        [2]     VciValue,
        egressPeakCellRateCLP0Plus1                     [3]     INTEGER,
        egressPeakCellRateCLP0                          [4]     INTEGER,
        ingressPeakCellRateCLP0Plus1                    [5]     INTEGER,
        ingressPeakCellRateCLP0                         [6]     INTEGER,
        egressSustainableCellRateCLP0Plus1              [7]     INTEGER,
        egressSustainableCellRateCLP0                   [8]     INTEGER,
        ingressSustainableCellRateCLP0Plus1             [9]     INTEGER,
        ingressSustainableCellRateCLP0                  [10]    INTEGER,
        egressCDVToleranceCLP0Plus1                     [11]    INTEGER,
        egressCDVToleranceCLP0                          [12]    INTEGER,
        ingressCDVToleranceCLP0Plus1                    [13]    INTEGER,
        ingressCDVToleranceCLP0                         [14]    INTEGER,
        egressMaxBurstSizeCLP0Plus1                     [15]    INTEGER,
        egressMaxBurstSizeCLP0                          [16]    INTEGER,
        ingressMaxBurstSizeCLP0Plus1                    [17]    INTEGER,
        ingressMaxBurstSizeCLP0                         [18]    INTEGER,
        bufferRelease                                   [19]    BOOLEAN,
        maxCc                                           [20]    INTEGER,
        maxInformationFieldLength                       [21]    INTEGER,
        maxLengthSscopUuField                           [22]    INTEGER,
        maxPd                                           [23]    INTEGER,
        maxSscopCreditToPeer                            [24]    INTEGER,
        maxStat                                         [25]    INTEGER,
        sscopTimerCc                                    [26]    INTEGER,
        sscopTimerIdle                                  [27]    INTEGER,
        sscopTimerKeepAlive                             [28]    INTEGER,
        sscopTimerNoResponse                            [29]    INTEGER,
        sscopTimerPoll                                  [30]    INTEGER}

ProtocolType::= INTEGER {
        RTMC                    (1),
        BBCC                    (2)}

ListVb5InterfacesRequestInfo ::= SEQUENCE OF ObjectInstance

ListVb5InterfacesRequestResult ::= SEQUENCE OF INTEGER  -- Logical service port number

ListVcsFailureInfo ::= INTEGER {
        unspecified                     (0),
        unknownLspNumber                (1),
        unknownLupNumber                (2),
        unknownVpciValue                (3),
        unknownLupVpciCombination       (4)}

ListVb5VcsRequestInfo ::= SEQUENCE {
        logicalServicePortNumber        [0]     INTEGER,
        logicalUserPortNumber           [1]     INTEGER OPTIONAL,
        vpciValue                       [2]     VpciValue OPTIONAL}
```

```
ListVb5VcsRequestResult ::= CHOICE {
        success         [0]     ListVcsSuccessInfo,
        failure         [1]     ListVcsFailureInfo}

ListVcsSuccessInfo ::= SEQUENCE {
        logicalUserPortNumber   [0]     INTEGER OPTIONAL,
        vpciValue               [1]     VpciValue,
        vciValue                [2]     VciValue}

ListVb5VpsRequestInfo ::= SEQUENCE {
        logicalServicePortNumber[0]     INTEGER,
        logicalUserPortNumber   [1]     INTEGER OPTIONAL}

ListVb5VpsRequestResult ::= CHOICE {
        success         [0]     ListVb5VpsSuccessInfo,
        failure         [1]     ListVb5VpFailureInfo}

ListVb5VpFailureInfo ::= INTEGER {
        unspecified                     (0),
        unknownLspNumber                (1),
        unknownLupNumber                (2)}

ListVb5VpsSuccessInfo ::= SEQUENCE OF Vb5VpDetails

Vb5VpDetails ::= SEQUENCE {
        logicalUserPortNumber           [0]     INTEGER OPTIONAL,
        physicalPort                    [1]     INTEGER,
        vpiValue                        [2]     INTEGER,
        vpciValue                       [3]     INTEGER OPTIONAL,
        maxNumVciBitsNearEnd            [4]     INTEGER OPTIONAL,
        maxNumVciBitsSupported          [5]     INTEGER OPTIONAL,
        maxNumActiveVccsAllowed         [6]     INTEGER OPTIONAL,
        maxNumActiveVccsNearEnd         [7]     INTEGER OPTIONAL}

LoopNotAddedInfo ::= INTEGER {
        unspecified                     (0),
        unknownLogicalServicePort       (1),
        unknownLogicalUserPort          (2),
        unknownVpciValue                (3),
        unknownVciValue                 (4),
        loopAlreadyPresent              (5)}

LoopNotRemovedInfo ::= INTEGER {
        unspecified                     (0),
        unknownLogicalServicePort       (1),
        unknownLogicalUserPort          (2),
        unknownVpciValue                (3),
        unknownVciValue                 (4),
        noLoopPresent                   (5)}

RemoveAnLoopRequestInfo ::= AddAnLoopRequestInfo

RemoveAnLoopRequestResult ::= CHOICE {
        loopRemoved             [0]     NULL,
        loopNotRemoved          [1]     LoopNotRemovedInfo}

RemoveLupsIndicationInfo ::= RemoveLupsRequestInfo

RemoveLupsRequestInfo ::= AddLupsRequestInfo
```

```
RemoveLupsRequestResult ::= INTEGER {
        lupRemoved (0),
        lupNotRemoved (1)}

RemoveVb5ConnectionIndicationInfo ::= RemoveVb5ConnectionRequestInfo

RemoveVb5ConnectionRequestInfo ::= SEQUENCE {
        vciValueA                       [0]     INTEGER OPTIONAL,
        vciValueB                       [1]     INTEGER OPTIONAL,
        physicalPortA                   [2]     INTEGER,
        vpiValueA                       [3]     INTEGER,
        logicalServicePortA             [4]     INTEGER OPTIONAL,
        vpciValueA                      [5]     INTEGER OPTIONAL,
        physicalPortB                   [6]     INTEGER,
        vpiValueB                       [7]     INTEGER,
        logicalServicePortB             [8]     INTEGER OPTIONAL,
        vpciValueB                      [9]     INTEGER OPTIONAL}

RemoveVb5ConnectionRequestResult ::= INTEGER {
        vb5ConnectionRemoved (0),
        vb5ConnectionNotRemoved (1)}

RemoveVb5InterfaceIndicationInfo ::= RemoveVb5InterfaceRequestInfo

RemoveVb5InterfaceRequestInfo ::= AddVb5InterfaceRequestInfo

RemoveVb5InterfaceRequestResult ::= INTEGER {
        vb5InterfaceRemoved (0),
        vb5InterfaceNotRemoved (1)}

RemoveVb5ProtocolIndicationInfo ::= RemoveVb5ProtocolRequestInfo

RemoveVb5ProtocolRequestInfo ::= SEQUENCE {
        logicalServicePortNumber[0]     INTEGER,
        vb5ProtocolType         [1]     Vb5ProtocolType}

RemoveVb5ProtocolRequestResult ::= INTEGER {
        vb5ProtocolRemoved (0),
        vb5ProtocolNotRemoved (1)}

RemoveVb5ProtocolVpIndicationInfo ::= RemoveVb5ProtocolVpRequestInfo

RemoveVb5ProtocolVpRequestInfo ::= SEQUENCE {
        logicalServicePortNumber[0]     INTEGER,
        vpciValue               [1]     VpciValue}

RemoveVb5ProtocolVpRequestResult ::= INTEGER {
        vb5ProtocolVpRemoved (0),
        vb5ProtocolVpNotRemoved (1)}

RemoveVb5VcsIndicationInfo ::= RemoveVb5VcsRequestInfo

RemoveVb5VcsRequestInfo::= AddVb5VcsRequestInfo

RemoveVb5VcsRequestResult ::= INTEGER {
        vb5VcsRemoved (0),
        vb5VcsNotRemoved (1)}

RemoveVb5VpsIndicationInfo ::= RemoveVb5VpsRequestInfo
```

```
RemoveVb5VpsRequestInfo ::= SEQUENCE {
        logicalServicePortNumber[0]     INTEGER,
        logicalServiceSubport    [1]    INTEGER OPTIONAL,
        removeVpInfo             [2]    RemoveVpInfo}

RemoveVb5VpsRequestResult ::= INTEGER {
        vb5VpsRemoved (0),
        vb5VpsNotRemoved (1)}

RemoveVpInfo ::= SEQUENCE OF SEQUENCE {
        physicalPort             [0]    INTEGER,
        vpiValue                 [1]    VpiValue,
        vpciValue                [2]    VpciValue}

ResourceStatusIndicationInfo ::= SEQUENCE {
        logicalServicePortNumber[0]     INTEGER,
        logicalUserPortNumber    [1]    INTEGER OPTIONAL,
        vpciValue                [2]    VpciValue,
        resourceStatus           [3]    ResourceStatus}

ResourceStatus ::= INTEGER {
        fullyOperational                          (0),
        administratelyBlockedTestCallsAllowed     (1),
        administratelyBlockedNoCellFlow           (2),
        fault                                     (3)}

SnAccessLabelsInquiryResult ::= SEQUENCE {
        snLabel          [0]    INTEGER OPTIONAL,
        interfaceLabel   [1]    INTEGER OPTIONAL}

SpecifiedNniVpci ::= SEQUENCE {
        logicalUserPortNumber    [0]    INTEGER OPTIONAL,
        vpciValue                [1]    VpciValue}

SpecifiedNniVpciOrRemoteVp ::= CHOICE {
        specifiedNniVpci         [0]    SpecifiedNniVpci,
        specifiedRemoteVp        [1]    SpecifiedVp}

SpecifiedVc ::= SEQUENCE {
        physicalPort     [0]    INTEGER,
        vpiValue         [1]    VpiValue,
        vciValue         [2]    VciValue}

SpecifiedVp ::= SEQUENCE {
        physicalPort     [0]    INTEGER,
        vpiValue         [1]    VpiValue}

SpecifiedVpOrVc ::= CHOICE {
        specifiedVp      [0]    SpecifiedVp,
        specifiedVc      [1]    SpecifiedVc}

Vb5ProtocolType ::= INTEGER {
        rtmc (0),
        bbcc (1)}

VpciNotAuditedInfo ::= INTEGER {
        unspecified (0),
        unknownLupNumber (1),
        unknownVpci (2),
        unknownPhysicalPort (3),
        unknownVpiValue (4)}
```

```
VpProfile ::= SEQUENCE {
        maxNumVciBitsNearEnd        [0]    INTEGER,
        maxNumVciBitsSupported      [1]    INTEGER,
        maxNumActiveVccsAllowed     [2]    INTEGER,
        maxNumActiveVccsNearEnd     [3]    INTEGER}

END – of Q832-3ASN1DefinedTypesModule
```

7 Protocol stacks

The protocol stacks specified in ITU-T Q.811, Q.812, G.773 and the SDH digital cross-connect part of ITU-T G.784 can be used as part of the protocol stack for this Recommendation. The following Recommendations should be used to extend these stacks to include ATM:

– Q.2811 Broadband Q3 and X interfaces – Lower Layer Protocols

– Q.2812 Broadband Q3 and X interfaces – Upper Layer Protocols

ANNEX A

Management requirements

The management requirements are given below and in the provisioning principles for VB5.1 and VB5.2 interfaces.

A.1 General requirements

The general requirements include the general management coordination functions between the access network and the service node across Q3/X interfaces.

A.1.1 Coordinated VP and VC configuration

The configuration management function must support the coordinated addition and removal of VPs and VCs at both the UNIs and at the VB5 interfaces so that VP and VCs can be added and removed without disruption.

A.1.2 VPC checking

A mechanism is required to check the identity of VPCs which are set up between a user port and a service node so that mistakes in the cross-connection within an access network can be identified.

A.1.3 Coordination of port configuration data

The coordination of configuration information relating to user ports and service ports and their VPs and VCs is required to ensure consistency between the access network and the service node.

A.1.4 Coordination of VPCI values

There is a requirement for management coordination between the SN and the AN concerning the allocation of VPCI values for connections.

A.1.5 Consistency of configuration

There is a requirement to check the consistency of the configuration of logical user ports, logical service ports, and UNI accesses.

A.1.6 Availability of information

Information concerning a VB5 interface should not be visible to operators other than those related by that VB5 interface.

A.2 Coordination of the VB5 interface

A.2.1 Creation

There is a requirement to coordinate the creation of the VB5 interface, with the following information:

1) VP identifiers;
2) VCCs allocated for B-BCC and RTMC-protocols;
3) VCI range;
4) maximum number of simultaneously active VCCs;
5) maximum bandwidth of the VPC, provided by the traffic descriptor, on both directions (egress and ingress) which specifies e.g.:
   - peak cell rate;
   - cell delay variation tolerance;
   - sustainable cell rate;
   - maximum burst size;
6) quality of service class of the VP;
7) other VCs provisioned in the VP;
8) the traffic profile of the VC, describing e.g.:
   - peak cell rate;
   - cell delay variation tolerance;
   - sustainable cell rate;
   - maximum burst size;
9) quality of service class of the VC.

A.2.2 Verification and auditing

There is a requirement to verify and audit the correct configuration of the VB5 interface (for information involved see creation).

A.2.3 Modification

There is a requirement to coordinate the modification of the VB5 interface.

A.2.4 Deletion

There is a requirement to coordinate the deletion of the VB5 interface.

A.2.5 VC and VP provisioning

There is a requirement to coordinate the provisioning of VCs and VPs for users, with the following information:

1) VP/VC identifiers;
2) VCI range for VPs;
3) maximum number of simultaneously active VCCs in VP;

4) maximum bandwidth of the VPC, provided by the traffic descriptor, on both directions (egress and ingress) which specifies e.g.:
   - peak cell rate;
   - cell delay variation tolerance;
   - sustainable cell rate;
   - maximum burst size;
5) quality of service class of the VP;
6) VCs provisioned in the VP;
7) the traffic profile of the VC, describing e.g.:
   - peak cell rate;
   - cell delay variation tolerance;
   - sustainable cell rate;
   - maximum burst size;

A.2.6 Verification and auditing

There is a requirement to verify and audit the correct configuration of VC's and VPs for users (for information involved see provisioning).

A.2.7 VP and VC modification

There is a requirement to coordinate the modification of VC's and VPs for users.

A.2.8 Verification and auditing

There is a requirement to verify and audit the correct configuration of the VB5 interface (for information involved see creation).

A.2.9 VP and VC deletion

There is a requirement to coordinate the deletion of VC's and VPs for users.

A.3 Coordination of the UNI interface

A.3.1 Creation

There is a requirement to coordinate the creation of user VPs and user VCs between the users and the AN, with the following information:

1) VP/VC identifiers;
2) VCI range;
3) maximum number of simultaneously active VCCs;
4) maximum bandwidth of the VPC, provided by the traffic descriptor, on both directions (egress and ingress) which specifies e.g.:
   - peak cell rate;
   - cell delay variation tolerance;
   - sustainable cell rate;
   - maximum burst size;
5) quality of service class of the VP;

6) the traffic profile of the VC, describing e.g.:
   - peak cell rate;
   - cell delay variation tolerance;
   - sustainable cell rate;
   - maximum burst size;

7) quality of service class of the VC.

A.3.2 Verification and auditing

There is a requirement to verify and audit the correct configuration of user VPs and user VCs between the users and the AN (for information involved see creation).

A.3.3 Modification

There is a requirement to coordinate the modification of user VPs and user VCs between the users and the AN.

A.3.4 Deletion

There is a requirement to coordinate the deletion of user VPs and user VCs between the users and the AN.

A.4 Broadband Bearer Connection Control coordination requirements

The requirements here are based on the need to support the VB5.2 B-BCC protocol [3].

The VPC/VCC at the VB5 and at the UNI interface may be selected both by the SN and by the AN, therefore both of them, the AN and the SN, must know the relevant information to do the CAC functions and select the right VPC/VC identifiers.

A.4.1 VCs at the VB5 interface

In order to support creation, deletion and modification of a VC at the VB5.2 interface under the control of B-BCC procedures the following information shall be available in the AN and the SN:

- the VPCs associated to the VB5.2 interface; the information associated to each VPC shall include:
  1) VCI range;
  2) maximum number of simultaneously active VCCs;
  3) maximum bandwidth of the VPC, provided by the traffic descriptor, on both directions (egress and ingress) which specifies e.g.:
     - peak cell rate;
     - cell delay variation tolerance;
     - sustainable cell rate;
     - maximum burst size;
  4) quality of service class of the VP;
  5) VCs allocated in the VP;

6) the traffic profile of the VC, describing e.g.:
   - peak cell rate;
   - cell delay variation tolerance;
   - sustainable cell rate;
   - maximum burst size;
7) quality of service class of the VC.

This information are available in the AN and SN, provided by the managed object classes modelling VP/VC connection points, defined in ITU-T I.751.

A.4.2 VCs at the UNI interface

In order to support creation, deletion and modification of VCs at the UNI interface under the control of B-BCC procedures, the following information shall be available in the AN and the SN:

- the VPCs associated to the LUP; the information associated to each VPC shall include:
  1) VCI range;
  2) maximum number of simultaneously active VCCs;
  3) maximum bandwidth of the VPC, provided by the traffic descriptor, on both directions (egress and ingress) which specifies e.g.:
     - peak cell rate;
     - cell delay variation tolerance;
     - sustainable cell rate;
     - maximum burst size;
  4) quality of service class of the VP;
  5) VCs allocated in the VP;
  6) the traffic profile of the VC, describing e.g.:
     - peak cell rate;
     - cell delay variation tolerance;
     - sustainable cell rate;
     - maximum burst size;
  7) quality of service class of the VC.

This information is available in the AN, provided by the managed object classes modelling VP/VC connection points, defined in ITU-T I.751. The same information shall be provided to the SN.

A.5 Fault and performance management

The following requirements have been identified for fault and performance management.

A.5.1 Fault reporting

Faults must be reported over the X interface when VB5 traffic interfaces are not operational.

A.5.2 Fault localization

The X interface must support the coordination activities that allow a Service Node and an Access Network to cooperate to locate the source of faults, for example when loopbacks are used.

ANNEX B

Transaction requirements

These transactions are for incorporation into implementation protocols. These have been defined using the Remote Procedure Call Specification [7] since this provides a simple way of formally specifying transactions that is compatible with CMIP but is not limited to it. This annex does not specify a mandatory RPC implementation since parameter definitions are not consolidated as the goal here is clarity, not efficiency, but the use of this Annex as an RPC implementation is not precluded.

```
/*****************************************************************************************************
*****************************************************************************************************
* Transaction Definitions
*****************************************************************************************************
*****************************************************************************************************/

program XVB5 {
version {
/* */
addAnLoopRequestAnswer AddAnLoopRequest (addAnLoopRequestCall) = 1
addLupsIndicationAnswer AddLupsIndication (addLupsIndicationCall) = 2
addLupsRequestAnswer AddLupsRequest (addLupsRequestCall) = 3
addVb5ProtocolVpIndicationAnswer AddVb5ProtocolVpIndication (addVb5ProtocolVpIndicationCall) = 4
addVb5ProtocolVpRequestAnswer AddVb5ProtocolVpRequest (addVb5ProtocolVpRequestCall) = 5
addVb5InterfaceRequestAnswer AddVb5InterfaceRequest (addVb5InterfaceRequestCall) = 6
addVb5InterfaceIndicationAnswer AddVb5InterfaceIndication (addVb5InterfaceIndicationCall) = 7
addVb5ProtocolIndicationAnswer AddVb5ProtocolIndication (addVb5ProtocolIndicationCall) = 8
addVb5ProtocolRequestAnswer AddVb5ProtocolRequest (addVb5ProtocolRequestCall) = 9
addVb5VcsIndicationAnswer AddVb5VcsIndication (addVb5VcsIndicationCall) = 10
addVb5VcsRequestAnswer AddVb5VcsRequest (addVb5VcsRequestCall) = 11
addVb5VpsIndicationAnswer AddVb5VpsIndication (addVb5VpsIndicationCall) = 12
addVb5VpsRequestAnswer AddVb5VpsRequest (addVb5VpsRequestCall) = 13
addVb5ConnectionIndicationAnswer AddVb5ConnectionIndication (addVb5ConnectionIndicationCall) = 14
addVb5ConnectionRequestAnswer AddVb5ConnectionRequest (addVb5ConnectionRequestCall) = 15
auditVb5ConnectionRequestAnswer AuditVb5ConnectionRequest (auditVb5ConnectionRequestCall) = 16
auditVb5VpciRequestAnswer AuditVb5VpciRequest (auditVb5VpciRequestCall) = 17
listLupsRequestAnswer ListLupsRequest (listLupsRequestCall) = 18
listVb5ProtocolDetailsRequestAnswer ListVb5ProtocolDetailsRequest (listVb5ProtocolDetailsRequestCall) = 19
listVb5InterfacesRequestAnswer ListVb5InterfacesRequest (listVb5InterfacesRequestCall) = 20
listVb5VcsRequestAnswer ListVb5VcsRequest (listVb5VcsRequestCall) = 21
listVb5VpsRequestAnswer ListVb5VpsRequest (listVb5VpsRequestCall) = 22
removeAnLoopRequestAnswer RemoveAnLoopRequest (removeAnLoopRequestCall) = 23
removeLupsIndicationAnswer RemoveLupsIndication (removeLupsIndicationCall) = 24
removeLupsRequestAnswer RemoveLupsRequest (removeLupsRequestCall) = 25
removeVb5ProtocolVpIndicationAnswer RemoveVb5ProtocolVpIndication
(removeVb5ProtocolVpIndicationCall) = 26
removeVb5ProtocolVpRequestAnswer RemoveVb5ProtocolVpRequest (removeVb5ProtocolVpRequestCall) = 27
removeVb5InterfaceIndicationAnswer RemoveVb5InterfaceIndication (removeVb5InterfaceIndicationCall) = 28
removeVb5InterfaceRequestAnswer RemoveVb5InterfaceRequest (removeVb5InterfaceRequestCall) = 29
removeVb5ProtocolIndicationAnswer RemoveVb5ProtocolIndication (removeVb5ProtocolIndicationCall) = 30
removeVb5ProtocolRequestAnswer RemoveVb5ProtocolRequest (removeVb5ProtocolRequestCall) = 31
removeVb5VcsIndicationAnswer RemoveVb5VcsIndication (removeVb5VcsIndicationCall) = 32
removeVb5VcsRequestAnswer RemoveVb5VcsRequest (removeVb5VcsRequestCall) = 33
removeVb5VpsIndicationAnswer RemoveVb5VpsIndication (removeVb5VpsIndicationCall) = 34
removeVb5VpsRequestAnswer RemoveVb5VpsRequest (removeVb5VpsRequestCall) = 35
removeVb5ConnectionIndicationAnswer RemoveVb5ConnectionIndication
(removeVb5ConnectionIndicationCall) = 36
removeVb5ConnectionRequestAnswer RemoveVb5ConnectionRequest (removeVb5ConnectionRequestCall) = 37
anServiceLabelInquiryAnswer AnServiceLabelInquiry (anServiceLabelInquiryCall) = 38
```

```
snAccessLabelsInquiryAnswer SnAccessLabelsInquiry (snAccessLabelsInquiryCall) = 39
resourceStatusIndicationAnswer ResourceStatusIndication (resourceStatusIndicationCall) = 40
/* */
} = 1  /* denotes version 1 */
} = 8323  /* denotes Q.832.3 */

/*****************************************************************************************************
*****************************************************************************************************
* Parameter Definitions
*****************************************************************************************************
*****************************************************************************************************/

/*****************************************************************************************************
* Parameter Definitions for Transaction 1
*****************************************************************************************************
This transaction is used by the OS of the SN to request the OS of the AN to loop a connection so that cells sent to
the AN will be returned.
*/
struct addAnLoopRequestCall {
        unsigned int LogicalServicePort;
        union switch (bool VpcReachesUserPort) {
                case TRUE:
                        unsigned int LogicalUserPort;
                case FALSE:
                        void;
        } OptionalUserPort;
        unsigned int VpciValue;
        union switch (bool VccLoopback) {
                case TRUE:
                        unsigned int VciValue;
                case FALSE:
                        void;
        } OptionalVciValue;
};
struct addAnLoopRequestAnswer {
        union switch (bool Failure) {
                case TRUE:
                        enum {
                                Unspecified                     = 0,
                                UnknownLogicalServicePort       = 1,
                                UnknownLogicalUserPort          = 2,
                                UnknownVpciValue                = 3,
                                UnknownVciValue                 = 4,
                                LoopAlreadyPresent              = 5
                        } FailureReason;
                case FALSE:
                        void;
        } OptionalFailureReason;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 2
*****************************************************************************************************
This transaction is used to notify the peer OS of the addition of logical user ports to a VB5 interface.
*/
struct addLupsIndicationCall {
        unsigned int LogicalServicePortNumber;
        unsigned int LogicalUserPortNumber<>;
};
struct addLupsIndicationAnswer {
        bool Acknowledge;
};
```

```
/*****************************************************************************************************
* Parameter Definitions for Transaction 3
*****************************************************************************************************
This transaction is used to request the peer OS to add logical user ports to a VB5 interface.
*/
struct addLupsRequestCall {
        unsigned int LogicalServicePortNumber;
        unsigned int LogicalUserPortNumber<>;
};
struct addLupsRequestAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 4
*****************************************************************************************************
This transaction is used to notify the peer OS of the addition of a protocol VP to a VB5 interface.
*/
struct addVb5ProtocolVpIndicationCall {
        unsigned int LogicalServicePortNumber;
        unsigned int VpciValue;
        union switch (bool ProfileSupported) {
                case TRUE:
                        struct {
                                unsigned int maxNumVciBitsNearEnd;
                                unsigned int maxNumVciBitsSupported;
                                unsigned int maxNumActiveVccsAllowed;
                                unsigned int maxNumActiveVccsNearEnd;
                        } Profile;
                case FALSE:
                        void;
        } OptionalProfile
};
struct addVb5ProtocolVpIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 5
*****************************************************************************************************
This transaction is used to request the peer OS to add a protocol VP to a VB5 interface.
*/
struct addVb5ProtocolVpRequestCall {
        unsigned int LogicalServicePortNumber;
        unsigned int VpciValue;
        union switch (bool ProfileSupported) {
                case TRUE:
                        struct {
                                unsigned int maxNumVciBitsNearEnd;
                                unsigned int maxNumVciBitsSupported;
                                unsigned int maxNumActiveVccsAllowed;
                                unsigned int maxNumActiveVccsNearEnd;
                        } Profile;
                case FALSE:
                        void;
        } OptionalProfile
};
struct addVb5ProtocolVpRequestAnswer {
        bool Acknowledge;
};
```

```
/*****************************************************************************************************
Parameter Definitions for Transaction 6
*****************************************************************************************************
This transaction is used to inform the peer OS that a new VB5 interface has been added.
*/
struct addVb5InterfaceIndicationCall {
        unsigned int LogicalServicePortNumber;
};
struct addVb5InterfaceIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 7
*****************************************************************************************************
This transaction is used to request the peer OS to add a VB5 interface.
*/
struct addVb5InterfaceRequestCall {
        unsigned int LogicalServicePortNumber;
};
struct addVb5InterfaceRequestAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 8
*****************************************************************************************************
This transaction is used to notify the peer OS of the addition of a protocol to an existing VB5 interface.
*/
struct addVb5ProtocolIndicationCall {
        unsigned int LogicalServicePortNumber;
        unsigned int VpciValue;
        unsigned int VciValue;
        enum {
                RTMC        = 1,
                BBCC        = 2
        } ProtocolType;
        struct {
                        /* Cell Rate is in cells per second. */
                        /* Cell Delay Variation is in microseconds. */
                        /* Burst Size is in cells. */
                unsigned int EgressPeakCellRateCLP0Plus1;
                unsigned int EgressPeakCellRateCLP0;
                unsigned int IngressPeakCellRateCLP0Plus1;
                unsigned int IngressPeakCellRateCLP0;
                unsigned int EgressSustainableCellRateCLP0Plus1;
                unsigned int EgressSustainableCellRateCLP0;
                unsigned int IngressSustainableCellRateCLP0Plus1;
                unsigned int IngressSustainableCellRateCLP0;
                unsigned int EgressCDVToleranceCLP0Plus1;
                unsigned int EgressCDVToleranceCLP0;
                unsigned int IngressCDVToleranceCLP0Plus1;
                unsigned int IngressCDVToleranceCLP0;
                unsigned int EgressMaxBurstSizeCLP0Plus1;
                unsigned int EgressMaxBurstSizeCLP0;
                unsigned int IngressMaxBurstSizeCLP0Plus1;
                unsigned int IngressMaxBurstSizeCLP0;
        } QosParameters;
        struct {
                bool BufferRelease;         /* on connection release */
                unsigned int MaxCc;         /* maximum value in PDUs of the state variable VT(CC) */
                unsigned int MaxInformationFieldLength;     /* in octets */
                unsigned int MaxLengthSscopUuField;         /* in octets */
```

```
            unsigned int MaxPd;         /* maximum value in PDUs of the state variable VT(PD) */
            unsigned int MaxSscopCreditToPeer;          /* absolute value in PDUs of the receive window */
            unsigned int MaxStat;       /* maximum number of list elements in a STAT PDU */
            unsigned int SscopTimerCc;      /* time interval between transmissions in milliseconds */
            unsigned int SscopTimerIdle;    /* idle phase in milliseconds */
            unsigned int SscopTimerKeepAlive;           /* keep alive phase in milliseconds */
            unsigned int SscopTimerNoResponse;          /* no response time in milliseconds */
            unsigned int SscopTimerPoll;    /* active POLL phase in milliseconds */
        } SaalParameters;
};
struct addVb5ProtocolIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 9
*****************************************************************************************************
This transaction is used to request the peer OS to add a protocol to an existing VB5 interface.
*/
struct addVb5ProtocolRequestCall {
        unsigned int LogicalServicePortNumber;
        unsigned int VpciValue;
        unsigned int VciValue;
        enum {
                RTMC        = 1,
                BBCC        = 2
        } ProtocolType;
        struct {
                        /* Cell Rate is in cells per second. */
                        /* Cell Delay Variation is in microseconds. */
                        /* Burst Size is in cells. */
                unsigned int EgressPeakCellRateCLP0Plus1;
                unsigned int EgressPeakCellRateCLP0;
                unsigned int IngressPeakCellRateCLP0Plus1;
                unsigned int IngressPeakCellRateCLP0;
                unsigned int EgressSustainableCellRateCLP0Plus1;
                unsigned int EgressSustainableCellRateCLP0;
                unsigned int IngressSustainableCellRateCLP0Plus1;
                unsigned int IngressSustainableCellRateCLP0;
                unsigned int EgressCDVToleranceCLP0Plus1;
                unsigned int EgressCDVToleranceCLP0;
                unsigned int IngressCDVToleranceCLP0Plus1;
                unsigned int IngressCDVToleranceCLP0;
                unsigned int EgressMaxBurstSizeCLP0Plus1;
                unsigned int EgressMaxBurstSizeCLP0;
                unsigned int IngressMaxBurstSizeCLP0Plus1;
                unsigned int IngressMaxBurstSizeCLP0;
        } QosParameters;
        struct {
                bool BufferRelease;         /* on connection release */
                unsigned int MaxCc;         /* maximum value in PDUs of the state variable VT(CC) */
                unsigned int MaxInformationFieldLength;     /* in octets */
                unsigned int MaxLengthSscopUuField;         /* in octets */
                unsigned int MaxPd;         /* maximum value in PDUs of the state variable VT(PD) */
                unsigned int MaxSscopCreditToPeer;          /* absolute value in PDUs of the receive window */
                unsigned int MaxStat;       /* maximum number of list elements in a STAT PDU */
                unsigned int SscopTimerCc;      /* time interval between transmissions in milliseconds */
                unsigned int SscopTimerIdle;    /* idle phase in milliseconds */
                unsigned int SscopTimerKeepAlive;           /* keep alive phase in milliseconds */
                unsigned int SscopTimerNoResponse;          /* no response time in milliseconds */
                unsigned int SscopTimerPoll;    /* active POLL phase in milliseconds */
        } SaalParameters;
};
```

```
struct addVb5ProtocolRequestAnswer {
        bool Acknowledge;

/*****************************************************************************************************
* Parameter Definitions for Transaction 10
*****************************************************************************************************
This transaction is used to notify the peer OS of the addition of VCs to a VP which is associated with a VB5
interface.
*/
struct addVb5VcsIndicationCall {
        unsigned int LogicalServicePortNumber;
        unsigned int *LogicalServiceSubport;
                /* present if VPC reaches a user port */
        unsigned int VpciValue;
        unsigned int VciValue<>;
};
struct addVb5VcsIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 11
*****************************************************************************************************
This transaction is used to request the peer OS to add VCs to a VP which is associated with a VB5 interface.
*/
struct addVb5VcsRequestCall {
        unsigned int LogicalServicePortNumber;
        unsigned int *LogicalServiceSubport;
                /* present if VPC reaches a user port */
        unsigned int VpciValue;
        unsigned int VciValue<>;
};
struct addVb5VcsRequestAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 12
*****************************************************************************************************
This transaction is used to notify the peer OS of the addition of VPs that are associated with a VB5 interface.
*/
struct addVb5VpsIndicationCall {
        unsigned int LogicalServicePortNumber;
        unsigned int *LogicalServiceSubport;
                /* present if VPC reaches a user port */
        struct {
                unsigned int PhysicalPort;
                unsigned int VpiValue;
                unsigned int VpciValue;
                union switch (bool ProfileSupported) {
                        case TRUE:
                                struct {
                                        unsigned int maxNumVciBitsNearEnd;
                                        unsigned int maxNumVciBitsSupported;
                                        unsigned int maxNumActiveVccsAllowed;
                                        unsigned int maxNumActiveVccsNearEnd;
                                } Profile;
                        case FALSE:
                                void;
                } OptionalProfile
        } VpInfo<>;
};
struct addVb5VpsIndicationAnswer {
```

```
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 13
*****************************************************************************************************
This transaction is used to request the peer OS to add VPs that are associated with a VB5 interface.
*/
struct addVb5VpsRequestCall {
        unsigned int LogicalServicePortNumber;
        unsigned int *LogicalServiceSubport;
                /* present if VPC reaches a user port */
        struct {
                unsigned int PhysicalPort;
                unsigned int VpiValue;
                unsigned int VpciValue;
                union switch (bool ProfileSupported) {
                        case TRUE:
                                struct {
                                        unsigned int maxNumVciBitsNearEnd;
                                        unsigned int maxNumVciBitsSupported;
                                        unsigned int maxNumActiveVccsAllowed;
                                        unsigned int maxNumActiveVccsNearEnd;
                                } Profile;
                        case FALSE:
                                void;
                } OptionalProfile
        } VpInfo<>;
};
struct addVb5VpsRequestAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 14
*****************************************************************************************************
This transaction is used to notify the peer OS of the addition of a VP or a VC cross connection associated with a
VB5 interface. The egress direction is out of the Access Network towards Service Node. The ingress direction is
into the Access Network from the Service Node.
*/
struct addVb5ConnectionIndicationCall {
                /* Cell Rate is in cells per second. */
                /* Cell Delay Variation is in microseconds. */
                /* Burst Size is in cells. */
        unsigned int EgressPeakCellRateCLP0Plus1;
        unsigned int EgressPeakCellRateCLP0;
        unsigned int IngressPeakCellRateCLP0Plus1;
        unsigned int IngressPeakCellRateCLP0;
        unsigned int EgressSustainableCellRateCLP0Plus1;
        unsigned int EgressSustainableCellRateCLP0;
        unsigned int IngressSustainableCellRateCLP0Plus1;
        unsigned int IngressSustainableCellRateCLP0;
        unsigned int EgressCDVToleranceCLP0Plus1;
        unsigned int EgressCDVToleranceCLP0;
        unsigned int IngressCDVToleranceCLP0Plus1;
        unsigned int IngressCDVToleranceCLP0;
        unsigned int EgressMaxBurstSizeCLP0Plus1;
        unsigned int EgressMaxBurstSizeCLP0;
        unsigned int IngressMaxBurstSizeCLP0Plus1;
        unsigned int IngressMaxBurstSizeCLP0;
        enum {
                Class0          = 0,    /* best effort without specified parameters */
                Class1          = 1,    /* parameters as specified for CBR */
```

```
            Class2      = 2,    /* parameters as specified for VBR */
            Class3      = 3,    /* parameters as specified for connection-oriented data */
            Class4      = 4     /* parameters as specified for connectionless data */
            } EgressQosClass;
        enum {
            Class0      = 0,    /* best effort without specified parameters */
            Class1      = 1,    /* parameters as specified for CBR */
            Class2      = 2,    /* parameters as specified for VBR */
            Class3      = 3,    /* parameters as specified for connection-oriented data */
            Class4      = 4     /* parameters as specified for connectionless data */
            } IngressQosClass;
        union switch (bool VcCrossConnection) {
            case TRUE:
                struct {
                    unsigned int VciValueA;   /* VCI on side A */
                    unsigned int VciValueB;   /* VCI on side B */
                    } VciPoints;
            case FALSE:
                void;
            } ConnectionTypeDetails;
        unsigned int PhysicalPortA;
        unsigned int VpiValueA;
        union switch (bool DetailsForPortA) {
            case TRUE:
                struct {
                    unsigned int *LogicalServicePort;
                    unsigned int *VpciValue;
                        /* cannot be specified for both ports */
                    } Details;
            case FALSE:
                void;
            } OptionalDetailsForPortA;
        unsigned int PhysicalPortB;
        unsigned int VpiValueB;
        union switch (bool DetailsForPortB) {
            case TRUE:
                struct {
                    unsigned int *LogicalUserPort;
                    unsigned int *VpciValue;
                        /* cannot be specified for both ports */
                    } Details;
            case FALSE:
                void;
            } OptionalDetailsForPortB;
        } AddVb5ConnectionIndicationCall;
};
struct addVb5ConnectionIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************
* Parameter Definitions for Transaction 15
*****************************************************************************************
This transaction is used to request the peer OS to add a VP or a VC cross connection associated with a VB5
interface. The egress direction is out of the Access Network towards Service Node. The ingress direction is into the
Access Network from the Service Node.
*/
struct addVb5ConnectionRequestCall {
            /* Cell Rate is in cells per second. */
            /* Cell Delay Variation is in microseconds. */
            /* Burst Size is in cells. */
        unsigned int EgressPeakCellRateCLP0Plus1;
        unsigned int EgressPeakCellRateCLP0;
```

```
        unsigned int IngressPeakCellRateCLP0Plus1;
        unsigned int IngressPeakCellRateCLP0;
        unsigned int EgressSustainableCellRateCLP0Plus1;
        unsigned int EgressSustainableCellRateCLP0;
        unsigned int IngressSustainableCellRateCLP0Plus1;
        unsigned int IngressSustainableCellRateCLP0;
        unsigned int EgressCDVToleranceCLP0Plus1;
        unsigned int EgressCDVToleranceCLP0;
        unsigned int IngressCDVToleranceCLP0Plus1;
        unsigned int IngressCDVToleranceCLP0;
        unsigned int EgressMaxBurstSizeCLP0Plus1;
        unsigned int EgressMaxBurstSizeCLP0;
        unsigned int IngressMaxBurstSizeCLP0Plus1;
        unsigned int IngressMaxBurstSizeCLP0;
        enum {
                Class0          = 0,    /* best effort without specified parameters */
                Class1          = 1,    /* parameters as specified for CBR */
                Class2          = 2,    /* parameters as specified for VBR */
                Class3          = 3,    /* parameters as specified for connection-oriented data */
                Class4          = 4     /* parameters as specified for connectionless data */
                } EgressQosClass;
        enum {
                Class0          = 0,    /* best effort without specified parameters */
                Class1          = 1,    /* parameters as specified for CBR */
                Class2          = 2,    /* parameters as specified for VBR */
                Class3          = 3,    /* parameters as specified for connection-oriented data */
                Class4          = 4     /* parameters as specified for connectionless data */
                } IngressQosClass;
        union switch (bool VcCrossConnection) {
                case TRUE:
                        struct {
                                unsigned int VciValueA;   /* VCI on side A */
                                unsigned int VciValueB;   /* VCI on side B */
                                } VciPoints;
                case FALSE:
                        void;
                } ConnectionTypeDetails;
        unsigned int PhysicalPortA;
        unsigned int VpiValueA;
        union switch (bool DetailsForPortA) {
                case TRUE:
                        struct {
                                unsigned int *LogicalServicePort;
                                unsigned int *VpciValue;
                                        /* cannot be specified for both ports */
                                } Details;
                case FALSE:
                        void;
                } OptionalDetailsForPortA;
        unsigned int PhysicalPortB;
        unsigned int VpiValueB;
        union switch (bool DetailsForPortB) {
                case TRUE:
                        struct {
                                unsigned int *LogicalUserPort;
                                unsigned int *VpciValue;
                                        /* cannot be specified for both ports */
                                } Details;
                case FALSE:
                        void;
                } OptionalDetailsForPortB;
        } AddVb5ConnectionRequestCall;
};
```

```
struct addVb5ConnectionRequestAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 16
*****************************************************************************************************
This transaction is used to request the peer OS to audit a connection which is associated with a VB5 interface.
*/
struct auditVb5ConnectionRequestCall {
        unsigned int LogicalServicePortNumber;
        union switch (bool VpCrossConnection) {
                case TRUE:
                        struct {
                                unsigned int PhysicalPort;
                                unsigned int VpiValue;
                        } KnownVpEnd;
                case FALSE:
                        struct {
                                unsigned int PhysicalPort;
                                unsigned int VpiValue;
                                unsigned int VciValue;
                        } KnownVcEnd;
        } KnownEnd;
};
struct auditVb5ConnectionRequestAnswer {
        union switch (bool Success) {
                case FALSE:
                        enum {
                                Unspecified             = 0,
                                UnknownPhysicalPort     = 1,
                                UnknownVpiValue         = 2,
                                UnknownVciValue         = 3
                        } FailureReason;
                case TRUE:
                        union switch (bool VpCrossConnection) {
                                case TRUE:
                                        struct {
                                                unsigned int PhysicalPort;
                                                unsigned int VpiValue;
                                        } OtherVpEnd;
                                case FALSE:
                                        struct {
                                                unsigned int PhysicalPort;
                                                unsigned int VpiValue;
                                                unsigned int VciValue;
                                        } OtherVcEnd;
                        } OtherEnd;
        } AuditVb5ConnectionRequestAnswer;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 17
*****************************************************************************************************
This transaction is used to request the peer OS to audit a VPCI which is associated with a VB5 interface.
*/
struct auditVb5VpciRequestCall {
        unsigned int LogicalServicePortNumber;
        union switch (bool KnownVpci) {
                case TRUE:
                        struct {
                                unsigned int *LogicalUserPortNumber;
                                        /* present if VPC reaches a user port */
```

```
                        unsigned int VpciValue;
                    } KnownNniVpci;
                case FALSE:
                    struct {
                        unsigned int PhysicalPort;
                        unsigned int VpiValue;
                    } KnownRemoteVp;
        } KnownEnd;
};
struct auditVb5VpciRequestAnswer {
        union switch (bool Success) {
                case FALSE:
                    enum {
                        Unspecified             = 0,
                        UnknownLupNumber        = 1,
                        UnknownVpci             = 2,
                        UnknownPhysicalPort     = 3,
                        UnknownVpiValue         = 4
                    } FailureReason;
                case TRUE:
                    union switch (bool ReturnedVpci) {
                        case TRUE:
                            struct {
                                unsigned int *LogicalUserPortNumber;
                                    /* present if VPC reaches user port */
                                unsigned int VpciValue
                            } ReturnedNniVpci;
                        case FALSE:
                            struct {
                                unsigned int PhysicalPort;
                                unsigned int VpiValue;
                            } ReturnedRemoteVp;
                    } ReturnedEnd;
        } AuditVb5VpciRequestAnswer;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 18
*****************************************************************************************************
This transaction is used to request the peer OS to list the logical user ports associated with a VB5 interface
between an Access Network and a Service Node which the two Operations Systems together control.
*/
struct listLupsRequestCall {
        unsigned int LogicalServicePortNumber;
};
struct listLupsRequestAnswer {
        union switch (bool Success) {
                case FALSE:
                    enum {
                        Unspecified             = 0,
                        UnknownLspNumber        = 1
                    } FailureReason;
                case TRUE:
                    unsigned int LogicalUserPortNumber<>;
        } ListLupsRequestAnswer;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 19
*****************************************************************************************************
This transaction is used to request the peer OS to list the details of the protocols of a VB5 interface between an
Access Network and a Service Node which the two Operations Systems together control.
*/
```

```
struct listVb5ProtocolDetailsRequestCall {
        unsigned int LogicalServicePortNumber;
};
struct listVb5ProtocolDetailsRequestAnswer {
        union switch (bool Success) {
                case FALSE:
                        enum {
                                Unspecified             = 0,
                                UnknownLspNumber        = 1
                        } FailureReason;
                case TRUE:
                        struct {
                                enum {
                                        RTMC        = 1,
                                        BBCC        = 2
                                        } ProtocolType;
                                unsigned int VpciValue;
                                unsigned int VciValue;
                                struct {
                                                /* Cell Rate is in cells per second. */
                                                /* Cell Delay Variation is in microseconds. */
                                                /* Burst Size is in cells. */
                                        unsigned int EgressPeakCellRateCLP0Plus1;
                                        unsigned int EgressPeakCellRateCLP0;
                                        unsigned int IngressPeakCellRateCLP0Plus1;
                                        unsigned int IngressPeakCellRateCLP0;
                                        unsigned int EgressSustainableCellRateCLP0Plus1;
                                        unsigned int EgressSustainableCellRateCLP0;
                                        unsigned int IngressSustainableCellRateCLP0Plus1;
                                        unsigned int IngressSustainableCellRateCLP0;
                                        unsigned int EgressCDVToleranceCLP0Plus1;
                                        unsigned int EgressCDVToleranceCLP0;
                                        unsigned int IngressCDVToleranceCLP0Plus1;
                                        unsigned int IngressCDVToleranceCLP0;
                                        unsigned int EgressMaxBurstSizeCLP0Plus1;
                                        unsigned int EgressMaxBurstSizeCLP0;
                                        unsigned int IngressMaxBurstSizeCLP0Plus1;
                                        unsigned int IngressMaxBurstSizeCLP0;
                                } QosParameters;
                                struct {
                                        bool BufferRelease;/* on connection release */
                                        unsigned int MaxCc;
                                                /* maximum value in PDUs of the state variable VT(CC) */
                                        unsigned int MaxInformationFieldLength;/* in octets */
                                        unsigned int MaxLengthSscopUuField;     /* in octets */
                                        unsigned int MaxPd;
                                                /* maximum value in PDUs of the state variable VT(PD) */
                                        unsigned int MaxSscopCreditToPeer;
                                                /* absolute value in PDUs of the receive window */
                                        unsigned int MaxStat;
                                                /* maximum number of list elements in a STAT PDU */
                                        unsigned int SscopTimerCc;
                                                /* time interval between transmissions in milliseconds */
                                        unsigned int SscopTimerIdle;     /* idle phase in milliseconds */
                                        unsigned int SscopTimerKeepAlive;
                                                /* keep alive phase in milliseconds */
                                        unsigned int SscopTimerNoResponse;
                                                /* no response time in milliseconds */
                                        unsigned int SscopTimerPoll;
                                                /* active POLL phase in milliseconds */
                                } SaalParameters;
                        } ProtocolDetails<>;
        } ListProtocolDetailsRequestAnswer;
```

```
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 20
*****************************************************************************************************
This transaction is used to request the peer OS to list the identities of the VB5 interfaces between Access
Network(s) and the Service Node(s) which the two Operations Systems together control.
*/
struct listVb5InterfacesRequestCall {
        void;
};
struct listVb5InterfacesRequestAnswer {
        unsigned int LogicalServicePortNumber<>;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 21
*****************************************************************************************************
This transaction is used to request the peer OS to list the VCs associated with a VB5 interface.
*/
struct listVb5VcsRequestCall {
        unsigned int LogicalServicePortNumber;
        unsigned int *LogicalUserPortNumber;
                /* present if response constrained to a particular LUP */
        unsigned int *VpciValue;
                /* present if response constrained to a particular VPCI */
};
struct listVb5VcsRequestAnswer {
        union switch (bool Success) {
                case FALSE:
                        enum {
                                Unspecified                     = 0,
                                UnknownLspNumber                = 1,
                                UnknownLupNumber                = 2,
                                UnknownVpciValue                = 3,
                                UnknownLupVpciCombination       = 4
                        } FailureReason;
                case TRUE:
                        struct {
                                unsigned int *LogicalUserPort;
                                        /* present if VCC reached LUP */
                                unsigned int VpciValue;
                                unsigned int VciValue<>;
                        } VpciVcs<>;
        } ListVb5VcsRequestAnswer;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 22
*****************************************************************************************************
This transaction is used to request the peer OS to list the VPs associated with a VB5 interface.
*/
struct listVb5VpsRequestCall {
        unsigned int LogicalServicePortNumber;
        union switch (bool AllVps)
                case TRUE:
                        void;
                case FALSE:
                        unsigned int *LogicalUserPortNumber;
                                /* absent for selection of VPCs terminating in the AN */
        } Filter;
        unsigned int *LogicalUserPortNumber;
};
```

```
struct listVb5VpsRequestAnswer {
        union switch (bool Success) {
                case FALSE:
                        enum {
                                Unspecified                     = 0,
                                UnknownLspNumber                = 1,
                                UnknownLupNumber                = 2
                        } FailureReason;
                case TRUE:
                        struct
                                unsigned int *LogicalUserPort;
                                        /* present if VPC reaches LUP */
                                struct {
                                        unsigned int PhysicalPort;
                                        unsigned int VpiValue;
                                        unsigned int VpciValue;
                                        union switch (bool ProfileSupported) {
                                                case TRUE:
                                                        struct {
                                                                unsigned int maxNumVciBitsNearEnd;
                                                                unsigned int maxNumVciBitsSupported;
                                                                unsigned int maxNumActiveVccsAllowed;
                                                                unsigned int maxNumActiveVccsNearEnd;
                                                        } Profile;
                                                case FALSE:
                                                        void;
                                        } OptionalProfile
                                } VpInfo;
                        } SubportVps<>;
        } ListVb5VpsRequestAnswer;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 23
*****************************************************************************************************
This transaction is used by the OS of the SN to request the OS of the AN to remove a loop from a connection.
*/
struct removeAnLoopRequestCall {
        unsigned int LogicalServicePort;
        union switch (bool VpcReachesUserPort) {
                case TRUE:
                        unsigned int LogicalUserPort;
                case FALSE:
                        void;
        } OptionalUserPort;
        unsigned int VpciValue;
        union switch (bool VccLoopback) {
                case TRUE:
                        unsigned int VciValue;
                case FALSE:
                        void;
        } OptionalVciValue;
};
struct removeAnLoopRequestAnswer {
        union switch (bool Failure) {
                case TRUE:
                        enum {
                                Unspecified                     = 0,
                                UnknownLogicalServicePort       = 1,
                                UnknownLogicalUserPort          = 2,
                                UnknownVpciValue                = 3,
                                UnknownVciValue                 = 4,
                                NoLoopPresent                   = 5
```

```
                } FailureReason;
            case FALSE:
                void;
        } OptionalFailureReason;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 24
*****************************************************************************************************
This transaction is used to notify the peer OS of the removal of logical user ports from a VB5 interface.
*/
struct removeLupsIndicationCall {
        unsigned int LogicalServicePortNumber;
        unsigned int LogicalUserPortNumber<>;
};
struct removeLupsIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 25
*****************************************************************************************************
This transaction is used to request the peer OS to remove logical user ports from a VB5 interface.
*/
struct removeLupsRequestCall {
        unsigned int LogicalServicePortNumber;
        unsigned int LogicalUserPortNumber<>;
};
struct removeLupsRequestAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 26
*****************************************************************************************************
This transaction is used to notify the peer OS of the removal of the protocol VP from a VB5 interface.
*/
struct removeVb5ProtocolVpIndicationCall {
        unsigned int LogicalServicePortNumber;
        unsigned int VpciValue;
};
struct removeVb5ProtocolVpIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 27
*****************************************************************************************************
This transaction is used to request the peer OS to remove the protocol VP from a VB5 interface.
*/
struct removeVb5ProtocolVpRequestCall {
        unsigned int LogicalServicePortNumber;
        unsigned int VpciValue;
};
struct removeVb5ProtocolVpRequestAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 28
*****************************************************************************************************
This transaction is used to notify the peer OS of the removal of a VB5 interface.
*/
```

```
struct removeVb5InterfaceIndicationCall {
        unsigned int LogicalServicePortNumber;
};
struct removeVb5InterfaceIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 29
*****************************************************************************************************
This transaction is used to request the peer OS to remove a VB5 interface.
*/
struct removeVb5InterfaceRequestCall {
        unsigned int LogicalServicePortNumber;
};
struct removeVb5InterfaceRequestAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 30
*****************************************************************************************************
This transaction is used to notify the peer OS of the removal of VCs from a VP which is associated with a VB5
interface.
*/
struct removeVb5VcsIndicationCall {
        unsigned int LogicalServicePortNumber;
        unsigned int *LogicalServiceSubport;
                /* present if VPC reaches a user port */
        unsigned int VpciValue;
        unsigned int VciValue<>;
};
struct removeVb5VcsIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 31
*****************************************************************************************************
This transaction is used to request the peer OS to remove VCs from a VP which is associated with a VB5
interface.
*/
struct removeVb5VcsRequestCall {
        unsigned int LogicalServicePortNumber;
        unsigned int *LogicalServiceSubport;
                /* present if VPC reaches a user port */
        unsigned int VpciValue;
        unsigned int VciValue<>;
};
struct removeVb5VcsRequestAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 32
*****************************************************************************************************
This transaction is used to notify the peer OS of the removal of a protocol from a VB5 interface.
*/
struct removeVb5ProtocolIndicationCall {
        unsigned int LogicalServicePortNumber;
        enum {
                RTMC        = 1,
                BBCC        = 2
```

```
        } ProtocolType;
};
struct removeVb5ProtocolIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 33
*****************************************************************************************************
This transaction is used to request the peer OS to remove a protocol from a VB5 interface.
*/
struct removeVb5ProtocolRequestCall {
        unsigned int LogicalServicePortNumber;
        enum {
                RTMC        = 1,
                BBCC        = 2
        } ProtocolType;
};
struct removeVb5ProtocolRequestAnswer {
        bool Acknowledge;

/*****************************************************************************************************
* Parameter Definitions for Transaction 34
*****************************************************************************************************
This transaction is used to notify the peer OS of the removal of VPs that are associated with a VB5 interface.
*/
struct removeVb5VpsIndicationCall {
        unsigned int LogicalServicePortNumber;
        unsigned int *LogicalServiceSubport;
                /* present if VPC reaches a user port */
        struct {
                unsigned int PhysicalPort;
                unsigned int VpiValue;
                unsigned int VpciValue;
        } VpInfo<>;
};
struct removeVb5VpsIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 35
*****************************************************************************************************
This transaction is used to request the peer OS to remove VPs that are associated with a VB5 interface.
*/
struct removeVb5VpsRequestCall {
        unsigned int LogicalServicePortNumber;
        unsigned int *LogicalServiceSubport;
                /* present if VPC reaches a user port */
        struct {
                unsigned int PhysicalPort;
                unsigned int VpiValue;
                unsigned int VpciValue;
        } VpInfo<>;
};
struct removeVb5VpsRequestAnswer {
        bool Acknowledge;
};
```

```
/*****************************************************************************************************
* Parameter Definitions for Transaction 36
*****************************************************************************************************
This transaction is used to notify the peer OS of the removal of a VP or a VC cross connection associated with a
VB5 interface. The egress direction is out of the Access Network towards Service Node. The ingress direction is
into the Access Network from the Service Node.
*/
struct removeVb5ConnectionIndicationCall {
        union switch (bool VcCrossConnection) {
                case TRUE:
                        struct {
                                unsigned int VciValueA;   /* VCI on side A */
                                unsigned int VciValueB;   /* VCI on side B */
                                } VciPoints;
                case FALSE:
                        void;
                } ConnectionTypeDetails;
        unsigned int PhysicalPortA;
        unsigned int VpiValueA;
        union switch (bool DetailsForPortA) {
                case TRUE:
                        struct {
                                unsigned int *LogicalServicePort;
                                unsigned int *VpciValue;
                                        /* cannot be specified for both ports */
                                } Details;
                case FALSE:
                        void;
                } OptionalDetailsForPortA;
        unsigned int PhysicalPortB;
        unsigned int VpiValueB;
        union switch (bool DetailsForPortB) {
                case TRUE:
                        struct {
                                unsigned int *LogicalUserPort;
                                unsigned int *VpciValue;
                                        /* cannot be specified for both ports */
                                } Details;
                case FALSE:
                        void;
                } OptionalDetailsForPortB;
        } RemoveVb5ConnectionIndicationCall;
};
struct removeVb5ConnectionIndicationAnswer {
        bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 37
*****************************************************************************************************
This transaction is used to request the peer OS to remove a VP or a VC cross connection associated with a VB5
interface. The egress direction is out of the Access Network towards Service Node. The ingress direction is into the
Access Network from the Service Node.
*/
struct removeVb5ConnectionRequestCall {
        union switch (bool VcCrossConnection) {
                case TRUE:
                        struct {
                                unsigned int VciValueA;   /* VCI on side A */
                                unsigned int VciValueB;   /* VCI on side B */
                                } VciPoints;
                case FALSE:
                        void;
```

```
        } ConnectionTypeDetails;
    unsigned int PhysicalPortA;
    unsigned int VpiValueA;
    union switch (bool DetailsForPortA) {
        case TRUE:
            struct {
                unsigned int *LogicalServicePort;
                unsigned int *VpciValue;
                    /* cannot be specified for both ports */
                } Details;
        case FALSE:
            void;
        } OptionalDetailsForPortA;
    unsigned int PhysicalPortB;
    unsigned int VpiValueB;
    union switch (bool DetailsForPortB) {
        case TRUE:
            struct {
                unsigned int *LogicalUserPort;
                unsigned int *VpciValue;
                    /* cannot be specified for both ports */
                } Details;
        case FALSE:
            void;
        } OptionalDetailsForPortB;
    } RemoveVb5ConnectionRequestCall;
};
struct removeVb5ConnectionRequestAnswer {
    bool Acknowledge;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 38
*****************************************************************************************************
This transaction is used by the OS of an AN to request the label that an SN uses for the AN.
*/
struct anServiceLabelInquiryCall {
    void;
};
struct anServiceLabelInquiryAnswer {
    union switch (bool AnLabelKnown) {
    case TRUE:
        unsigned int AnLabel;
    case FALSE:
        void;
    } AnServiceLabelInquiryAnswer;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 39
*****************************************************************************************************
This transaction is used by the OS of an SN to request the label that an AN uses for the SN and the local reference
that the AN uses for the VB5 interface.
*/
struct snAccessLabelsInquiryCall {
    void;
};
struct snAccessLabelsInquiryAnswer {
    union switch (bool SnAndIntLabelsKnown) {
    case TRUE:
        unsigned int SnLabel;
        unsigned int InterfaceLabel;
    case FALSE:
```

```
            void;
        } snAccessLabelsInquiryAnswer;
};

/*****************************************************************************************************
* Parameter Definitions for Transaction 40
*****************************************************************************************************
This transaction is used to notify the peer OS of the change of status of a Resource.
*/
struct resourceStatusIndicationCall {
        unsigned int LogicalServicePort;
        union switch (bool SpecifiedLogicalUserPort) {
                case TRUE:
                        unsigned int LogicalServicePort;
                case FALSE:
                        void;
                } OptionalLogicalServicePort;
        unsigned int VpciValue<>;
        enum {
                FullyOperational                        = 0,
                AdministratelyBlockedTestCallsAllowed   = 1,
                AdministratelyBlockedNoCellFlow         = 2,
                Fault                                   = 3
        } Status;
};
struct resourceStatusIndicationAnswer {
        bool Acknowledge;
};
```

APPENDIX I

Bibliography

[1] ITU-T G.773 (1993), *Protocol suites for Q-interfaces for management of transmission systems*.

[2] ITU-T G.774 (2001), *Synchronous digital hierarchy (SDH) management information model for the network element view.*

[3] ITU-T G.803 (2000), *Architecture of transport networks based on the synchronous digital hierarchy (SDH)*.

[4] ITU-T I.211 (1993), *B-ISDN service aspects.*

[5] ITU-T I.311 (1996), *B-ISDN general network aspects.*

[6] ITU-T I.327 (1993), *B-ISDN functional architecture*.

[7] ITU-T I.356 (2000), *B-ISDN ATM layer cell transfer performance*.

[8] ITU-T I.371 (2000), *Traffic control and congestion control in B-ISDN*.

[9] ITU-T I.413 (1993), *B-ISDN user-network interface*.

[10] ITU-T I.432.x family, *B-ISDN user-network interface – Physical layer specification*.

[11] ITU-T I.580 (1995), *General arrangements for interworking between B-ISDN and 64 kbit/s based ISDN*.

[12] ITU-T I.610 (1999), *B-ISDN operation and maintenance principles and functions*.

[13] ITU-T I.732 (2000), *Functional characteristics of ATM equipment*.

[14] ITU-T M.3200 (1997), *TMN management services and telecommunications managed areas: overview*.

[15] ITU-T M.3207.1 (1996), *TMN management service: Maintenance aspects of B-ISDN management*.

[16] ITU-T M.3400 (2000), *TMN Management Functions*.

[17] ITU-T M.3610 (1996), *Principles for applying the TMN concept to the management of B-ISDN*.

[18] ITU-T Q.821 (2000), *Stage 2 and stage 3 description for the Q3 interface – Alarm Surveillance*.

[19] ITU-T Q.822 (1994), *Stage 1, stage 2 and stage 3 description for the Q3 interface – Performance management*.

[20] ITU-T X.208 (1988), *Specification of abstract syntax notification one (ASN.1)*.

[21] ITU-T X.701 (1997) | ISO/IEC 10040:1998, *Information technology – Open Systems Interconnection – System management overview*.

[22] ITU-T X.722 (1992) | ISO/IEC 10165-4:1992, *Information technology – Open Systems Interconnection – Structure of management information: Guidelines for definition of managed objects*.

[23] ITU-T X.733 (1992) | ISO/IEC 10164-4:1992, *Information technology – Open Systems Interconnection – Systems Management: Alarm reporting function*.

[24] ITU-T X.734 (1992) | ISO/IEC 10164-5:1993, *Information technology – Open Systems Interconnection – Systems Management: Event report management function*.

[25] ITU-T X.735 (1992) | ISO/IEC 10164-6:1993, *Information technology – Open Systems Interconnection – Systems Management: Log control functions*.

[26] ITU-T X.737 (1995) | ISO/IEC 10164-14:1996, *Information technology – Open Systems Interconnection – Systems Management: Confidence and diagnostic test categories*.

[27] ITU-T X.738 (1993) | ISO/IEC 10164-13:1995, *Information technology – Open Systems Interconnection – Systems Management: Summarization function*.

[28] ITU-T X.739 (1993) | ISO/IEC 10164-11:1994, *Information technology – Open Systems Interconnection – Systems Management: Metric objects and attributes*.

[29] ITU-T X.745 (1993) | ISO/IEC 10164-12:1994, *Information technology – Open Systems Interconnection – Systems Management: Test management function*.

[30] ITU-T X.746 (1995) | ISO/IEC 10164-15:1995, *Information technology – Open Systems Interconnection – Systems Management: Scheduling function*.

[31] ATM Forum Specification, *M4 Interface Requirements and Logical MIB, ATM Network Element View, Version 1.0*.

[32] ATM Forum Specification, *CMIP Specification for the M4 Interface, Version 1.0*.

[33] ATM Forum Specification, *ATM User-Network Interface Specification, Version 3.0*.

[34] ATM Forum Specification, *ATM User-Network Interface Specification, Version 3.1*.

[35] ATM Forum Specification, *ATM User-Network Interface Specification, Version 4.0*.

APPENDIX II

Summary of transactions

Category	Name	Type	Origin
VB5 Creation	anServiceLabelInquiry	Action	AN
	snAccessLabelsInquiry	Action	SN
	addVb5InterfaceRequest	Action	both
	addVb5InterfaceIndication	Notification	both
VB5 Deletion	removeVb5InterfaceRequest	Action	both
	removeVb5InterfaceIndication	Notification	both
Modification – VB5 Protocol VP	addVb5ProtocolVpRequest	Action	both
	addVb5ProtocolIndication	Notification	both
	removeVb5ProtocolVpRequest	Action	both
	removeVb5ProtocolVpIndication	Notification	both
Modification – VB5 Protocols	addVb5ProtocolVpRequest	Action	both
	addVb5ProtocolVpIndication	Notification	both
	removeVb5ProtocolRequest	Action	both
	removeVb5ProtocolIndication	Notification	both
Modification – Logical User Ports	addLupsRequest	Action	both
	addLupsIndication	Notification	both
	removeLupsRequest	Action	both
	removeLupsIndication	Notification	both
Modification – VPs	addVb5VpsRequest	Action	both
	addVb5VpsIndication	Notification	both
	removeVb5VpsRequest	Action	both
	removeVb5VpsIndication	Notification	both
Modification – VCs	addVb5VcsRequest	Action	both
	addVb5VcsIndication	Notification	both
	removeVb5VcsRequest	Action	both
	removeVb5VcsIndication	Notification	both
Modification – Connections	addVb5ConnectionRequest	Action	both
	addVb5ConnectionIndication	Notification	both
	removeVb5ConnectionRequest	Action	both
	removeVb5ConnectionIndication	Notification	both
Auditing – List Requests	listVb5InterfacesRequest	Action	both
	listVb5ProtocolDetailsRequest	Action	both
	listLupsRequest	Action	both
	listVb5VpsRequest	Action	both
	listVb5VcsRequest	Action	both
Auditing – Audit Requests	auditVb5ConnectionRequest	Action	both
	auditVb5VpciRequest	Action	both
Fault Reporting	resourceStatusIndication	Notification	both
Fault Localization	addAnLoopRequest	Action	SN
	removeAnLoopRequest	Action	SN

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	**Switching and signalling**
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure and Internet protocol aspects
Series Z	Languages and general software aspects for telecommunication systems

Geneva, 2001