INTERNATIONAL STANDARD

ISO/IEC 9041-1

Second edition
1997-06-15

Information technology — Open Systems Interconnection — Virtual Terminal Basic Class Protocol —

Part 1: Specification

Technologies de l'information — Interconnexion de systèmes ouverts (OSI) — Protocole de classe de base de terminal virtuel —

Partie 1: Spécification

Reference number
ISO/IEC 9041-1:1997(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO and IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

International Standard ISO 9041-1 was prepared by Joint Technical Committee ISO/IEC JTC1, *Information technology*, Subcommitee SC21, *Open systems interconnection, data management and open distributed processing*.

This second edition cancels and replaces the first edition (ISO/IEC 9041-1:1990), which has been technically revised. It also incorporates Amendment 2:1992, Technical Corrigendum 1:1992 and Technical Corrigendum 2:1993.

ISO/IEC 9041 consists of the following parts, under the general title *Information technology – Open Systems Interconnection – Virtual Terminal Basic Class Protocol*:

– Part 1: Specification

– Part 2: Implementation Conformance Statement (PICS) Proforma

Annexes A and B form an integral part of this part of ISO/IEC 9041.

Introduction

ISO/IEC 9041 is one of a set of International Standards produced to facilitate the interconnection of computer systems. It is related to other International Standards in the set as defined in the Reference Model for Open Systems Interconnection (ISO/IEC 7498-1). The Reference Model subdivides the area of standardization for interconnection into a series of layers of specification, each of manageable size.

This part of ISO/IEC 9041 defines the manner in which two protocol machines (called Virtual Terminal Protocol Machines or VTPMs) in the Application Layer of the Reference Model for Open Systems Interconnection communicate in order to provide the Virtual Terminal Basic Class Service defined in ISO/IEC 9040 making use of the Presentation Layer and of the association control service of ACSE within the Application Layer.

Part 2 of ISO/IEC 9041 includes the Protocol Implementation Conformance Statement (PICS) proforma for the Virtual Terminal Basic Class Protocol as defined in this part of ISO/IEC 9041.

Information technology – Open Systems Interconnection – Virtual Terminal Basic Class Protocol – Part 1: Specification

1 Scope

This part of ISO/IEC 9041 specifies:

a) a set of procedures for the connection-oriented transfer of data, control and reference information between protocol machines which implement the functions of a provider of the Basic Class Virtual Terminal Service;

b) the two modes in which those procedures operate;

c) the structure of protocol elements used for the transfer of data and control information and the mapping of these protocol elements onto ACSE and lower layer services;

d) the means of negotiating the functional units to be used by the protocol machines and the parameters of the service;

e) the structure and mapping of protocol elements used for the transfer of data and control information.

The procedures are defined in terms of:

f) the interactions between Virtual Terminal Protocol Machines through the exchange of Virtual Terminal protocol elements;

g) the interactions between a Virtual Terminal Protocol Machine and the Virtual Terminal service user in the same system through the exchange of Virtual Terminal service primitives;

h) the interactions between a Virtual Terminal Protocol Machine and the ACSE and Presentation Service providers through the exchange of service primitives.

This part of ISO/IEC 9041 also specifies conformance requirements (see clause 13) for systems implementing these procedures. It does not contain tests which can be used to demonstrate this conformance.

These procedures are applicable to instances of communication between systems that support the Basic Class Virtual Terminal Service in the Application Layer of the Reference Model for Open Systems Interconnection and which wish to interconnect in an open systems environment.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9041. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9040 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 2022:1994, *Information technology – Character code structure and extension techniques (fourth edition).*

ISO/IEC 6429:1992, *Information technology – Control functions for coded character sets (third edition).*

ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.*

ISO/IEC 7498-3:1997, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing.*

ISO/IEC 8326:1996, *Information technology – Open System Interconnection – Session service definition (second edition).*

ISO/IEC 8649:1996, *Information technology – Open Systems Interconnection – Service definition for the Association Control Service Element (second edition).*

ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition.*

ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

ISO/IEC 8825-1:1994, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

ISO/IEC 9040:1997, *Information technology – Open Systems Interconnection – Virtual Terminal Basic Class Service.*

ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services.*

The International Register of Coded Character Sets to be used with Escape Sequences. [1)]

3 Definitions

3.1 Global OSI definitions

This part of ISO/IEC 9041 is based on the concepts developed in ISO/IEC 7498, and makes use of the following terms defined in it:

1) application entity;
2) application layer;
3) protocol data unit;
4) user element.

Definitions of the following terms are given in ISO/IEC 10731:

5) primitive;
6) confirm (primitive);
7) indication (primitive);
8) request (primitive);
9) response (primitive);
10) service provider;
11) service user.

3.2 Association Control Service Element definitions

This part of ISO/IEC 9041 makes use of the following terms defined in ISO/IEC 8649:

1) application association;
2) application context.

3.3 Presentation Layer Service Element definition

This part of ISO/IEC 9041 makes use of the term abstract syntax which is defined in ISO/IEC 8822.

3.4 Virtual Terminal Service definitions

This part of ISO/IEC 9041 uses the following definitions contained in ISO/IEC 9040:

1) A-mode, Asynchronous Mode;
2) access-rule;
3) Application VT-user;
4) Block;
5) character-box graphic element;
6) Context Control Object (CCO);
7) current-VTE;
8) draft-VTE;
9) Field;
10) Field Definition Control Object (FDCO);
11) Field Definition Record (FDR);
12) Field Element;
13) Field Entry Condition (FEC);
14) Field Entry Event (FEE);
15) Field Entry Instruction (FEI);
16) Field Entry Instruction Control Object (FEICO);
17) Field Entry Instruction Record (FEIR);
18) Field Entry Pilot Control Object (FEPCO);
19) Field Entry Pilot Record (FEPR);
20) Field Entry Reaction (FER);
21) full-VTE;
22) Initial Facility Set;
23) net-effecting;
24) Reference Information Object (RIO);
25) repertoire;
26) ripple.
27) S-mode, Synchronous Mode;
28) Terminal VT-user;
29) Transmission Policy Control Object (TPCO);
30) VT-association;
31) VT-context-value;
32) VT-environment (VTE);
33) VT-user;
34) VTE-parameter;

3.5 Virtual Terminal Protocol definitions

For the purposes of this part of ISO/IEC 9041 the following definitions apply.

3.5.1 protocol element: An abstract unit of information, defined in clause 6, communicated between peer VTPMs, that maps

a) directly to an ACSE or presentation service primitive; and/or

b) to a PDU type that is conveyed by a user information parameter of ACSE or a user data parameter of the presentation service.

3.5.2 initiating VTPM: The VTPM that initiates an individual procedure.

3.5.3 target VTPM: The VTPM to which the protocol element initiating a procedure is directed.

3.5.4 sending VTPM: The initiating VTPM for a data transfer procedure.

3.5.5 receiving VTPM: The target VTPM for a data transfer procedure.

1) Available from the European Computer Manufacturers Association (ECMA), 114 Rue du Rhône, CH 1204 Genève, Switzerland.

3.5.6 dynamic conformance requirements: All those requirements (and options) which determine what observable behaviour is permitted in instances of communication.

3.5.7 static conformance requirements: Constraints which facilitate interworking by defining the requirements for kernel sets of capabilities of an implementation.

3.5.8 protocol implementation conformance statement (PICS): A statement made by the supplier of an implementation which states the capabilities and options which have been implemented, and any features which have been omitted.

3.5.9 VT-token: A single entity which maps onto all of the available session tokens provided by the Presentation Layer. If there are no session tokens available then both sides are considered to hold this token.

4 Abbreviations

4.1 Virtual Terminal Service abbreviations

A-mode	Asynchronous Mode
ACS	Access Control Store
CCA	Conceptual Communication Area
CCO	Context Control Object
CDS	Conceptual Data Store
CO	Control Object
CSS	Control, Signal and Status Store
DO	Display Object
DSD	Data Structure Definition
FDCO	Field Definition Control Object
FDR	Field Definition Record
FEC	Field Entry Condition
FEE	Filed Entry Event
FEI	Field Entry Instruction
FEICO	Field Entry Instruction Control Object
FEIR	Field Entry Instruction Record
FEPCO	Field Entry Pilot Control Object
FEPR	Field Entry Pilot Record
FER	Field Entry Reaction
MIN	Multiple Interaction Negotiation
RIO	Reference Information Object
S-mode	Synchronous Mode
TPCO	Transmission Policy Control Object
VT	Virtual Terminal
VTE	Virtual Terminal Environment
VTS	Virtual Terminal Service

4.2 Miscellaneous

ACSE	Association Control Service Element
ASN.1	Abstract Syntax Notation One
PAB	Provider Abort
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
TWA	Two-Way Alternate
TWS	Two-Way Simultaneous
VTP	Virtual Terminal Protocol
VTPM	Virtual Terminal Protocol Machine

4.3 VT Protocol elements

APQ	VT-P-ABORT
ASQ	VT-ASSOCIATE-REQ
ASR	VT-ASSOCIATE-RESP
AUQ	VT-U-ABORT
BKQ	VT-BREAK-REQ
BKR	VT-BREAK-RESP
DAQ	VT-ACK-RECEIPT
DLQ	VT-DELIVER
ENQ	VT-END-NEG-REQ
ENR	VT-END-NEG-RESP
EXQ	VT-P-EXCEPTION-REQ
EXR	VT-P-EXCEPTION-RESP
GTQ	VT-GIVE-TOKEN
HDQ	VT-HIGH-PRI-DATA
NAQ	VT-NEG-ACCEPT
NDQ	VT-DATA
NIQ	VT-NEG-INVITE
NJQ	VT-NEG-REJECT
NOQ	VT-NEG-OFFER
RLQ	VT-RELEASE-REQ
RLR	VT-RELEASE-RESP
RTQ	VT-REQUEST-TOKEN
SNQ	VT-START-NEG-REQ
SNR	VT-START-NEG-RESP
SPQ	VT-SWITCH-PROFILE-REQ
SPR	VT-SWITCH-PROFILE-RESP
UDQ	VT-URGENT-DATA

5 Overview

5.1 Virtual Terminal Service summary

The protocol specified in this part of ISO/IEC 9041 provides the Basic Class Virtual Terminal Service defined in ISO/IEC 9040. The VT-service primitives are listed in table 1.

Table 1 - Virtual Terminal Service Primitives

Service Primitive		Service Parameters
VT-ASSOCIATE	request indication	Called Application Entity Title Calling Application Entity Title VT-class VT-functional-units VT-mode VT-WAVAR-owner VT-profile-name VT-profile-arg-offer-list
	response confirm	Responding Application Entity Title VT-functional-units VT-WAVAR-owner VT-profile-arg-value-list VT-result VT-user-failure-reason (confirm only)
VT-P-ABORT	indication	VT-reason
VT-RELEASE	request indication	
	response confirm	VT-result VT-user-failure-reason VT-provider-failure-reason (confirm only)
VT-U-ABORT	request indication	VT-user-failure-reason
VT-DATA (excluding the two following cases)	request indication	VT-echo-now VT-start-entry VT-object-update VT-object-descriptor VT-object-data
VT-DATA (high priority Control Objects only)	request indication	VT-object-update VT-object-descriptor VT-object-data
VT-DATA (urgent priority Control Objects only)	request indication	VT-object-update VT-object-descriptor VT-object-data
VT-DELIVER	request indication	VT-ack-request
VT-ACK-RECEIPT	request indication	
VT-GIVE-TOKEN	request indication	
VT-REQUEST-TOKEN	request indication	

Table 1 (concluded)

Service Primitive		Service Parameters
VT-SWITCH-PROFILE	request indication response confirm	VT-profile-name VT-profile-arg-offer-list VT-object-retention-list VT-profile-arg-value-list VT-result VT-user-failure-reason VT-provider-failure-reason (confirm only) VT-object-retention-list
VT-START-NEG	request indication response confirm	VT-profile-name VT-profile-arg-offer-list VT-profile-arg-value-list VT-result VT-user-failure-reason VT-provider-failure-reason (confirm only)
VT-END-NEG	request indication response confirm	VT-vte-choice VT-vte-failure-allowed VT-object-retention-list VT-vte-choice VT-result VT-user-failure-reason VT-provider-failure-reason (confirm only) VT-object-retention-list
VT-NEG-INVITE	request indication	VT-param-ident-list
VT-NEG-OFFER	request indication	VT-param-offer-list
VT-NEG-ACCEPT	request indication	VT-param-value-list
VT-NEG-REJECT	request indication	VT-param-ident-list
VT-BREAK	request indication response confirm	VT-token VT-information VT-token VT-information
VT-P-EXCEPTION	indication	VT-exception-source VT-exception-type VT-information

5.2 Model

The Basic Class Virtual Terminal Protocol operates between two Virtual Terminal Protocol Machines (VTPMs) in the Application layer of the OSI model. Protocol elements are exchanged between these, using the services of Association Control and of the Presentation layer as defined in ISO/IEC 8649 and ISO/IEC 8822.

The VT service is modelled as a single Conceptual Communication Area which is accessible to two communicating service users and which contains all the necessary information to allow these VT-users to derive a consistent view of the Virtual Devices which constitute the Virtual Terminal. The CCA is partitioned into four sub-areas:

a) a conceptual data store (CDS) containing one or more display objects;

b) a control, signaling and status store (CSS) containing a number of control objects;

c) an access control store (ACS);

d) a data structure definition (DSD) which parametrically defines the structure of the CDS and CSS.

The CCA is conceptually accessed by VT-users via service primitives in which information is transferred to or from the VT-users. Each VTPM is modelled as having its own CCA, see figure 1. These two CCAs constitute the VT context referred to in the Basic Class VTS definition. Each VTPM may also maintain a reset context which preserves the contents of all DOs and COs as they existed at the time the current VTE was established. This is used to provide the VT-context-value after completion of the break procedure, see 8.8.

5.3 Service assumed from the Presentation Layer

The protocol specified in this part of ISO/IEC 9041 assumes the use of the Presentation Service defined in ISO/IEC 8822. Information is transferred to and from the Presentation service provider in the primitives listed in table 2.

5.4 Service assumed from ACSE

The protocol specified in this part of ISO/IEC 9041 assumes the use of the Association Control service defined in ISO/IEC 8649. Information is transferred to and from ACSE in the service primitives listed in table 3.

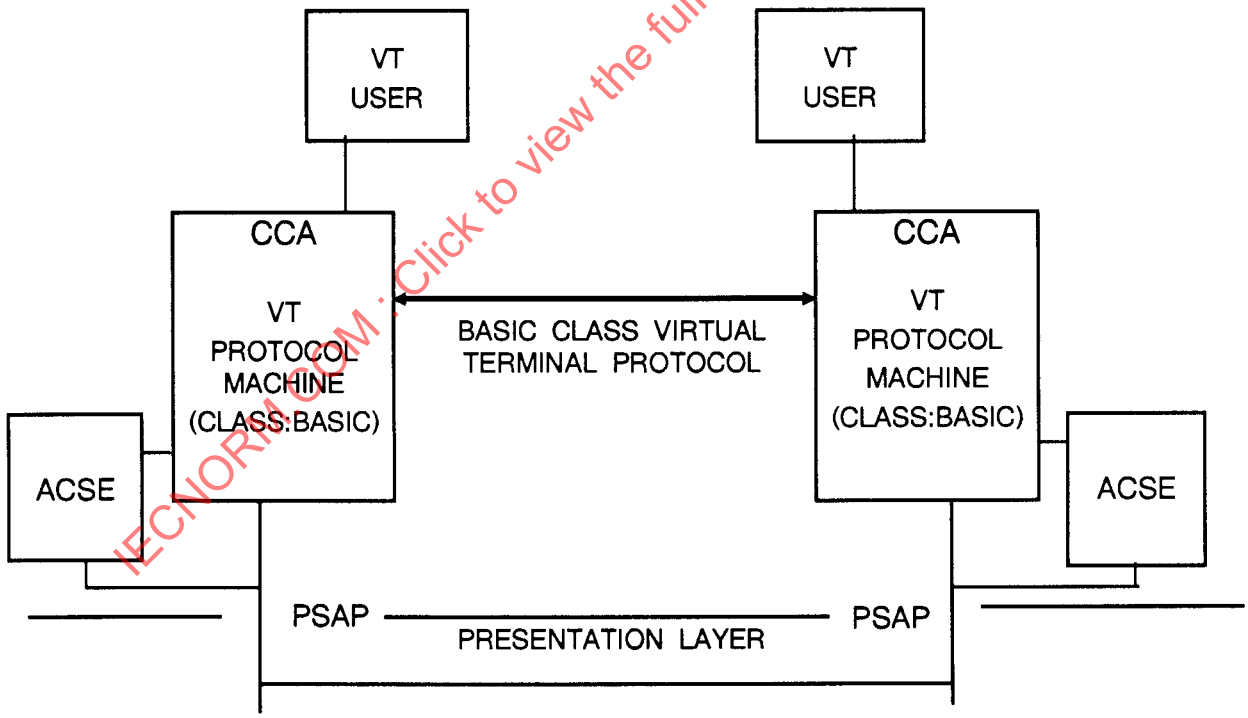

Figure 1 - Model

Table 2 - Presentation Service primitives

Presentation service	primitives
P-DATA	request indication
P-EXPEDITED-DATA	request indication
P-RESYNCHRONIZE	request indication response confirm
P-SYNCHRONIZE-MAJOR	request indication response confirm
P-TOKEN-GIVE	request indication
P-TOKEN-PLEASE	request indication
P-TYPED-DATA	request indication

Table 3 - ACSE service primitives

ACSE service	primitive
A-ASSOCIATE	request indication response confirm
A-RELEASE	request indication response confirm
A-ABORT	request indication
A-P-ABORT	indication

5.5 Functions of the VT Protocol

5.5.1 Association Establishment

The purpose of VT-association establishment is to establish a VT-association between two VT-users such that:

a) the use of this VT-protocol is confirmed;

b) the required dialogue discipline is agreed;

c) a suitable transfer syntax is used;

d) either an initial VTE is agreed or it is agreed that the VT-environment will be negotiated later before any data is transferred.

5.5.2 Association Termination

The purpose of VT-association Termination is to cause the VT-association to cease to exist.

5.5.3 Negotiation

Negotiation provides a mechanism by which the VT-users and VTPMs can agree upon the VT-environment to be used. Two forms of negotiation are supported, a single interaction profile switch and a multiple interaction dialogue.

5.5.4 Data Transfer

The purpose of Data Transfer is the sending, from one VTPM to the other, of structured data representing operations upon an object or objects in the CCA defined in the virtual terminal model.

5.5.5 Delivery Control

The purpose of Delivery Control is:

a) to allow the VT-user sending data to indicate points at which operations on VT objects conceptually take effect;

b) to allow the receipt of such a delivery point indication to be acknowledged;

c) to allow data which update objects with the same access rules to be grouped together and simultaneously be made available to the VT-user.

5.5.6 Dialogue Management

Dialogue Management provides the VT-users with access to the lower layer functions. This dialogue management enforces TWA discipline on the sending of some data and on the exercise of some control functions.

5.5.7 Error Handling

Error Handling permits detection of non-compliance with the protocol, reception of error indications from other service providers and the taking of appropriate action. The errors cause the association to be terminated with an appropriate error indication (identifying the source of error).

NOTE – The procedure for Error Handling is included with that for unconditional termination of an association.

5.5.8 Interrupt

Two interrupt functions are provided. The first is a non- destructive mechanism which allows priority information to be exchanged between the VT-users outside the normal data flow without disrupting that data flow. The second is a destructive mechanism which allows a VT-user unconditionally to bring the current dialogue to a halt. Information is exchanged between the VT-users to enable them to resynchronize their operations.

5.5.9 Exceptions Handling

Exceptions Handling permits the VT service provider to advise the VT-users of certain non-fatal exception conditions arising during the operation of a VT-association without terminating the VT-association.

5.6 Protocol functional units

This part of ISO/IEC 9041 defines the following functions which are available in the Basic Class Virtual Terminal Protocol independently of which functional units have been selected:

a) Association Establishment;
b) Association Termination;
c) Data Transfer;
d) Delivery Control (optional, see Note 1);
e) Dialogue Management (S-mode only);
f) Error Handling.

NOTE 1 – Use of delivery-control is controlled at the level of VTE-profile by a VTE-parameter.

This part of ISO/IEC 9041 also defines seven functional units of the Basic Class Virtual Terminal Protocol which correspond to the similarly named service functional units which are defined in ISO/IEC 9040. The following functions are available only if the corresponding functional unit has been selected:

g) Switch Profile Negotiation;
h) Multiple Interaction Negotiation;
i) Negotiated Release;
j) Break;
k) Urgent Data;
l) Reference Information Objects;
m) Exceptions.

The Multiple Interaction Negotiation functional unit may only be selected if the Switch Profile functional unit is also selected. The Exceptions functional unit may only be selected if the Break functional unit is also selected.

NOTE 2 – Service functional units defined in ISO/IEC 9040 but not included above do not affect the protocol procedures but may affect the allowed content of some protocol elements as defined in later clauses.

5.7 Modes of operation

The protocol operates in two modes: Asynchronous and Synchronous. In the Synchronous mode (S-mode) the presentation connection supporting the VT communication is treated as a TWA connection, and the VTPMs send normal update data to each other alternately. In the Asynchronous mode (A-mode) the presentation connection is treated as a TWS connection, and the VTPMs are not restricted in the exchange of data.

5.8 Access control

The initiation and reception of some protocol elements by a VT-user is conditional upon the location of the token and upon the access-rules associated with the objects affected by the elements.

6 Protocol elements

Table 4 lists the elements of the Virtual Terminal Basic Class protocol with the associated VT service and a cross reference to the clause where the protocol element is further described. The description includes, for each element, its purpose and a list of its parameters.

Table 4 - VT protocol elements

Protocol element	VT service	Cross reference
ASQ	VT-ASSOCIATE	6.2
ASR	VT-ASSOCIATE	6.3
APQ	VT-P-ABORT	6.1
AUQ	VT-U-ABORT	6.4
RLQ	VT-RELEASE	6.18
RLR	VT-RELEASE	6.19
HDQ	VT-DATA (high priority)	6.12
NDQ	VT-DATA (normal)	6.14
UDQ	VT-DATA (urgent priority)	6.25
DAQ	VT-ACK-RECEIPT	6.7
DLQ	VT-DELIVER	6.8
GTQ	VT-GIVE-TOKEN	6.11
RTQ	VT-REQUEST-TOKEN	6.20
BKQ	VT-BREAK	6.5
BKR	VT-BREAK	6.6
EXQ	VT-P-EXCEPTION	6.26
EXR	VT-P-EXCEPTION	6.27
SPQ	VT-SWITCH-PROFILE	6.23
SPR	VT-SWITCH-PROFILE	6.24
ENQ	VT-END-NEG	6.9
ENR	VT-END-NEG	6.10
NAQ	VT-NEG-ACCEPT	6.13
NIQ	VT-NEG-INVITE	6.15
NJQ	VT-NEG-REJECT	6.16
NOQ	VT-NEG-OFFER	6.17
SNQ	VT-START-NEG	6.21
SNR	VT-START-NEG	6.22

Procedures specifying the use of the elements are specified in clauses 7 through 10 and their mapping to the presentation service or ACSE and their ASN.1 structure are specified in clauses 11 and 12.

In some implementations it may not be possible to provide the parameters of the APQ and AUQ protocol elements. This is due to possible length limitations in the supporting services. Consequently, the decision whether to send these parameters is a local one based upon a knowledge of the capability of the supporting services. For the same reason the parameters of other protocol elements, while they may be sent, may be restricted as to their complexity. These latter are:

a) ASQ;
b) ASR;
c) SNQ;
d) SNR;
e) SPQ;
f) SPR;
g) UDQ.

NOTE – In the following sub-clauses the prefix VT- is used where there is a service parameter in ISO/IEC 9040 with the same name.

6.1 APQ (VT-P-ABORT)

6.1.1 Purpose

To force unconditional termination of the VT-association (initiated by the VTPM).

6.1.2 Parameter

VT-reason: Optional, if present takes one of the values "local error" or "protocol error".

6.2 ASQ (VT-ASSOCIATE-REQ)

6.2.1 Purpose

To request establishment of a VT-association.

6.2.2 Parameters

a) called-application-entity-title: Specifies the application entity with which a VT-association is to be established;

b) calling-application-entity-title: Identifies the application entity initiating the VT-association;

c) VT-class: Specifies the VT service class to be provided; takes the value "Basic" to invoke the service specified in ISO/IEC9040;

NOTE – The use of other values of this parameter for the invocation of other International Standards is outside the scope of ISO/IEC 9041.

d) VT-mode: Specifies whether A-mode or S-mode will be in effect when the association is established and whether mode switching is to be permitted;

e) VT-WAVAR-owner: Defines which side of the dialogue will initially own the WAVAR token if this exists; takes one of the values "initiator-side", "acceptor-side", or "acceptor-chooses"; omitted if WAVAR does not exist;

f) VT-profile-name: If present, the name of the profile (see ISO/IEC 9040); if absent, the default profile defined in ISO/IEC 9040 for A-mode or for S-mode, as selected by the VT-mode parameter, is used;

g) VT-profile-arg-offer-list: if present, a list of items in which each profile argument under negotiation is either a value, list of values or range of values; omitted if the profile identified by VT-profile-name has no profile arguments;

h) VT-functional-units: If present, specifies the proposed functional units of the current VT service class; takes one or more values from the following list:
   1) Profile Switch,
   2) Multiple Interaction Negotiation,
   3) Negotiated Release,
   4) Urgent Data,
   5) Destructive Break,
   6) Enhanced Access-rules,
   7) Structured Control Objects (Structured COs),
   8) Blocks,
   9) Fields,
   10) Reference Information Objects (RIOs),
   11) Ripple,
   12) Exceptions,
   13) Context Retention;

i) protocol-version: For the initiating VTPM this is a variable length bit string where each bit set to one indicates that the corresponding version of the protocol is supported. Bit 0 represents the first version, bit 1 represents the second, etc. Multiple bits may be set indicating support of multiple versions. No trailing bits higher than the highest version of this standard that the VTPM supports shall be sent, i.e., the last bit of the string must be set to one. The receiving VTPM shall ignore trailing bits higher than the one indicating the latest version of this part of ISO/IEC 9041 that it supports;

j) implementation-identifier: Optional, allows implementors to identify their implementation and its version for maintenance purposes.

6.3 ASR (VT-ASSOCIATE-RESP)

6.3.1 Purpose

To complete or refuse establishment of a VT-association.

6.3.2 Parameters

a) responding-application-entity-title: Identifies the application entity which is responding to a VT-association request;

b) VT-WAVAR-owner: As in 6.2 except that the value "acceptor chooses" is not allowed. Only present if the ASQ parameter was "acceptor chooses";

c) VT-profile-arg-value-list: If present, a list of parameters with unique values of each parameter selected from the list or range of values in the profile argument offer list; not present if the value of VT-result is "failure", see also 6.2.2 g);

d) VT-result: Takes one of the values "success", "success-with-warning" or "failure";

e) failure-reason: Conveys either a text string supplied by the responding VT-user or one of the values "VTE-incomplete", VTE-parameter-not-supported", "VTE-parameter-combination-not-supported", or "VTE-profile-not-supported";

f) protocol-version: In the case of successful establishment this is similar to the corresponding item in 6.2.2 except that only a single bit may be set to one. This bit selects the version of the protocol which will be used during the association. If the VT-result value is "failure" this bit string indicates the protocol version(s) of this part of ISO/IEC 9041 that can be supported by the rejecting VTPM;

g) VT-functional units: As in 6.2.2 h);

h) implementation-identifier: As in 6.2.2 j).

6.4 AUQ (VT-U-ABORT)

6.4.1 Purpose

To force unconditional termination of the VT-association (initiated by the VT-user).

6.4.2 Parameters

VT-user-failure-reason: optional; user provided text string.

6.5 BKQ (VT-BREAK-REQ)

6.5.1 Purpose

To request a destructive priority interrupt to be indicated to the remote VT-user by the remote VTPM.

6.5.2 Parameters

a) VT-WAVAR-owner: can take one of the values "initiator-side", "acceptor-side" or "acceptor-chooses"; omitted if WAVAR does not exist;

b) VT-information: Items as defined in ISO/IEC 9040.

6.6 BKR (VT-BREAK-RESP)

6.6.1 Purpose

To acknowledge to the remote VTPM that a destructive priority interrupt was indicated to the local VT-user.

6.6.2 Parameters

a) VT-WAVAR-owner: can take one of the values "initiator-side" or "acceptor-side"; omitted if WAVAR does not exist;

b) VT-information: Items as defined in ISO/IEC 9040.

6.7 DAQ (VT-ACK-RECEIPT)

6.7.1 Purpose

To acknowledge a delivery point identified by the receipt of a DLQ.

6.7.2 Parameters

None.

6.8 DLQ (VT-DELIVER)

6.8.1 Purpose

To designate delivery points in the stream of NDQ PDUs and, if required, to request acknowledgement of receipt.

6.8.2 Parameters

VT-ack-request: A single boolean indicating whether or not an acknowledgement is explicitly required.

6.9 ENQ (VT-END-NEG-REQ)

6.9.1 Purpose

To request the synchronized termination of a multiple interaction negotiation and to cause a transition to the data handling phase.

6.9.2 Parameters

a) VT-vte-choice: Takes one of the values "draft", "current" or "either";

b) VT-failure-allowed: Optional, if present takes one of the values "yes" or "no";

c) VT-object-retention-list: Optional, if present contains a list of DO and CO names whose contents are requested to be retained.

6.10 ENR (VT-END-NEG-RESP)

6.10.1 Purpose

To respond to a request to terminate negotiation.

6.10.2 Parameters

a) VT-vte-choice: Optional, if present takes one of the values "draft" or "current";

b) VT-result: Takes one of the values "success", "success-with-warning "or "failure" ;

c) VT-user-failure-reason: Optional user provided text string;

d) VT-provider-failure-reason: Optional, if present takes the value "VTE-incomplete";

e) vt-object-retention-list: Optional, if present contains a list of DO and CO names whose contents are agreed to be retained.

6.11 GTQ (VT-GIVE-TOKEN)

6.11.1 Purpose

To pass all defined session tokens to the peer VTPM.

6.11.2 Parameters

None.

6.12 HDQ (VT-HIGH-PRI-DATA)

6.12.1 Purpose

To convey updates to Control Objects for which the CO-priority VTE-parameter has the value "high".

6.12.2 Parameters

object updates: A sequence of zero or more data items. Each data item is a (CO-name, object update) pair. Each CO-name identifies a Control Object; the corresponding object update identifies permitted updates to that object. The permitted updates to Control Objects are defined in ISO/IEC 9040.

6.13 NAQ (VT-NEG-ACCEPT)

6.13.1 Purpose

To select values for one or more of the VTE-parameters specified in a previous VT-NEG-OFFER protocol element.

6.13.2 Parameters

VT-param-value-list: A list of VTE-parameters and corresponding values for those parameters selected from the alternatives specified in a previous VT-NEG-OFFER.

6.14 NDQ (VT-DATA)

6.14.1 Purpose

To update objects in a "controlled" manner and, for A-mode, to indicate if echoing or data entry may take place following processing of a data item.

6.14.2 Parameters

a) object updates: A sequence of zero or more data items. Each data item is a (object name, object update) pair. Each object name identifies a Display Object, Control Object or RIO; the corresponding object update identifies permitted updates to that object. The permitted updates to objects are defined in ISO/IEC 9040;

b) echo-now: If present, takes the value "echo";

c) start-entry :If present, takes the value "start-entry".

6.15 NIQ (VT-NEG-INVITE)

6.15.1 Purpose

To request the peer to provide values for specified VTE-parameters once an agreement to enter into multiple interaction negotiation has been reached.

6.15.2 Parameters

VT-param-ident-list: The names of VTE-parameters for which values are to be provided.

6.16 NJQ (VT-NEG-REJECT)

6.16.1 Purpose

To reject VTE-parameters contained in previous NOQs whose values and/or value ranges are unacceptable.

6.16.2 Parameters

VT-param-ident-list: The names of VTE-parameters whose values are being rejected.

6.17 NOQ (VT-NEG-OFFER)

6.17.1 Purpose

To pass a list of VTE-parameters and parameter values, lists of values, or value ranges to the peer once an agreement to enter into multiple interaction negotiation has been reached.

6.17.2 Parameters

VT-param-offer-list: A list of VTE-parameters and corresponding values, lists or ranges for those parameters.

6.18 RLQ (VT-RELEASE-REQ)

6.18.1 Purpose

To request the orderly termination of a VT-association.

6.18.2 Parameters

None.

6.19 RLR (VT-RELEASE-RESP)

6.19.1 Purpose

To accept or reject a request for orderly termination of a VT-association.

6.19.2 Parameters

a) VT-result: Takes one of the values "success" or "failure". The value "failure" may only be used if the Negotiated Release Session functional unit was successfully negotiated in the Session Requirements parameter of the A-ASSOCIATE, see 11.1.1;

b) failure-reason: Conveys either a text string provided by the responding VT-user or the value "collision-detected" when the failure is determined by the VTPM.

6.20 RTQ (VT-REQUEST-TOKEN)

6.20.1 Purpose

To request possession of all defined Session tokens.

6.20.2 Parameters

None.

6.21 SNQ (VT-START-NEG-REQ)

6.21.1 Purpose

To request the establishment of the Negotiation Active phase.

6.21.2 Parameters

a) VT-profile-name: As in 6.2.2 f);

b) VT-profile-arg-offer-list: As in 6.2.2 g).

6.22 SNR (VT-START-NEG-RESP)

6.22.1 Purpose

To indicate success or failure of the attempt to establish the Negotiation Active phase.

6.22.2 Parameters

a) VT-profile-arg-value-list: As in 6.3.2 c);

b) VT-result: Takes one of the values "success" or "failure";

c) failure-reason: Conveys either a text string supplied by the responding VT-user or one of the values "collision-detected" or "profile-not-supported".

6.23 SPQ (VT-SWITCH-PROFILE-REQ)

6.23.1 Purpose

To negotiate a switch to a new full-VTE constructed from a named VTE profile.

6.23.2 Parameters

a) VT-profile-name: As in 6.2.2 f);

b) VT-profile-arg-offer-list: As in 6.2.2 g);

c) VT-object-retention-list: if present contains a list of DO and CO names whose contents are requested to be retained.

6.24 SPR (VT-SWITCH-PROFILE-RESP)

6.24.1 Purpose

To indicate success or failure of the negotiation attempt.

6.24.2 Parameters

a) VT-profile-arg-value-list: As in 6.3.2 c);

b) VT-result: Takes one of the values "success" or "failure";

c) failure-reason: Conveys either a text string supplied by the responding VT-user or one of the values "collision-detected", "parameter-not-supported", "parameter-combination-not-supported" or "profile-not-supported";

d) VT-object-retention-list: if present contains a list of DO and CO names whose contents are agreed to be retained.

6.25 UDQ (VT-URGENT-DATA)

6.25.1 Purpose

To convey updates to Control Objects for which the CO-priority VTE-parameter has the value "urgent".

This protocol element is always available but its mapping onto lower layer services is dependent on the final value of the Session Requirements parameter, see 11.1.1.

6.25.2 Parameters

As in 6.12.2.

6.26 EXQ (VT-P-EXCEPTION-REQ)

6.26.1 Purpose:

To signal exception conditions between VTPMs.

6.26.2 Parameters

a) VT-exception-type: Takes one of the following symbolic values:

"RIO or temporary buffer full";

"too many fields";

"too many field elements";

"too many FERs";

"too many FEIs";

or a profile defined value of the form tag,value (INTEGER).

b) VT-information: Takes a value as defined in ISO/IEC 9040.

6.27 EXR (VT-P-EXCEPTION-RESP)

6.27.1 Purpose

To acknowledge an EXQ.

6.27.2 Parameters

None.

7 Procedures

The procedures of the Basic Class VTP are specified in Clauses 8 to 10 as sequences of actions performed by the peer VTPMs. The sequences are presented in tabular form, indicating

a) Step: numerical order in the sequence; when a sequence branches, the first step in each branch is suffixed with a different letter and subsequent steps in each branch retain this suffix (for example, if a branch occurs after step 4, the next steps are numbered 5a and 5b; 6a then follows 5a and 6b follows 5b);

b) VTPM: distinguishes the VTPM by its role in the procedure concerned (i.e., initiating-I, target-T, either-E, sending-S, or receiving-R);

c) Event: one of

1) receipt of the named service primitive from the VT–user,

2) receipt of the named protocol element from the peer VTPM,

3) the event PAB (see note 1)

4) an exception event internal to a VTPM (see note 2);

d) Action: specifies the action to be taken by the VTPM in response to the event.

NOTES

1 – The failure of some lower layer or some other service of the OSI model, which results in the loss of the VT-association, will be signaled to both VTPMs. Since such a signal is not passed between VTPMs it is not identified as a protocol element; it is however, a significant event and is identified by the name Provider Abort (PAB).

2 – An exception event internal to one VTPM is not inherently known to the other VTPM and is handled as defined in 8.8.

Conditions causing branches are included in the description of the event. Where an action concludes the sequence for an individual branch but is not the last one tabulated, the phrase "sequence ends" is appended. Information that is neither an event nor an action is placed within parentheses.

Each VTPM maintains a reset-context which was set from the current-VTE when it was established and from the initial CO and DO contents as defined in ISO/IEC 9040. This is used to provide the VT-context-value after completion of the break procedure, see 8.8 and table 15.

NOTES

3 – The value of reset-context is determined by the values negotiated for various VTE-parameters as defined in ISO/IEC 9040. No extra values need be explicitly stored.

4 – The reset-context is only used by the break procedure; it need not be maintained if the Break functional unit is not selected.

8 Primary procedures

This clause specifies the procedures to be used in terms of the sequences of actions to be performed by the communicating VTPMs. The procedures of this clause are to be used when no negotiation is permitted.

8.1 Association Establishment

Establishment of a VT-association, as required by a VT-ASSOCIATE request primitive, is effected by the procedures defined in table 5. On successful completion, any of the available facilities (i.e., data transfer, delivery control, token management and termination) may be used.

Table 6 defines the procedures for association refusal in terms of variant actions in place of those of table 5.

8.2 Agreed Release

The procedures defined in table 7 are used to release the VT-association. In S-mode a VTPM initiating a release action must hold the tokens. In A-mode collisions may occur between an attempted release and another attempted release or between an attempted release and a delivery action requiring confirmation. The refusal and collision portions of the table are each in the form of variations to be applied to the first portion.

8.3 Unconditional Termination

Unconditional termination of a VT-association may be initiated by a VT-user, a VTPM or by some supporting service. The procedures are defined in table 8.

8.4 Negotiation

The primary procedures do not include Negotiation.

8.5 Data Transfer

The Data Transfer procedures are defined in tables 9, 10, 11, 12, and 13. The procedures differ according to the type of data and the direction of transfer relative to the position of the tokens.

Table 5 - Association Establishment procedures

Step	VTPM	Event	Action
1	I	VT-ASSOCIATE request acceptable to initiating VTPM	Send ASQ.
2a	T	ASQ acceptable to target VTPM	Issue VT-ASSOCIATE indication
3aa	T	VT-ASSOCIATE response with VT-result = "success"	Send ASR with VT-result "success". Establish the "reset-context" according to the current contents of all COs and DOs.
4aaa	I	ASR with VT-result = "success" and ASR contents acceptable to initiating VTPM	Issue VT-ASSOCIATE confirm with VT-result="success". Establish the "reset-context" according to the current contents of all COs and DOs.

Table 6 - Association Refusal procedures

Step	VTPM	Event	Action
2b	T	ASQ not acceptable to target VTPM	Send ASR with VT-result="failure".
3ab	T	VT-ASSOCIATE response with VT-result = "failure"	Send ASR with the same VT-result value.
4ab/3b	I	ASR with VT-result = "failure"	Issue VT-ASSOCIATE confirm with the same VT-result value. Sequence ends.
4aab	I	ASR with VT-result = "success" and ASR contents not acceptable to initiating VTPM	Issue VT-ASSOCIATE confirm with VT-result="failure". Send APQ. Sequence ends.

8.6 Delivery Control

The Delivery Control procedures defined in table 13 apply if Simple Delivery Control or Quarantine Delivery Control has been negotiated for the current-VTE. The acknowledgement procedures only apply if specifically requested in the VT-DELIVER primitive.

Other service elements cause the forwarding and delivery actions to occur but do not cause a VT-DELIVER indication to be issued. The tables appropriate to these elements specify when this is to occur.

8.7 Token Management

The Token Management procedures allow a VTPM to pass the data token to the peer VTPM or to request the tokens from the peer VTPM. They apply to S-mode only. These procedures are defined in table 14.

8.8 Break and Exception

The VT–BREAK procedures provide the protocol facilities to support the VT–BREAK service, which is a destructive interrupt service available to either VT-user independently of the mode of operation or the ownership of access-rights.

The VT–EXCEPTION procedures provide the protocol facilities to support the VT–EXCEPTION service, which is a destructive interrupt service available to either VTPM independently of the mode of operation or the ownership of access-rights.

These procedures take precedence over other procedures and are defined in table 15.

Table 7 - Agreed Release procedures

Step	VTPM	Event	Action
Agreed Release with Tokens			
1a	I	VT-RELEASE request while holding tokens	Forward data as specified in table 9 or 10. Send RLQ.
2aa	T	RLQ without collision	Perform data delivery as specified in table 9 or 10. Issue a VT-RELEASE indication.
3aaa	T	VT-RELEASE response with VT-result = "success"	Forward data as specified in table 9 or 10. Send RLR with VT-result = "success".
4aaa	I	RLR with VT-result = "success"	Deliver data as specified in table 9 or 10. Issue VT-RELEASE confirm with VT-result = "success". Sequence ends.
Refusal of Release			
3aab	T	VT-RELEASE response with VT-result = "failure"	Send an RLR with VT-result = "failure".
4aab	I	RLR with VT-result = "failure"	Issue a VT-RELEASE confirm with VT-result = "failure". Do not perform data delivery. Sequence ends.
Release Collision			
2ab	T	RLQ when delivery acknowledgement is pending and negotiated release is available	Send RLR with VT-result="failure" and failure-reason = "collision-detected". Continue to await DAQ.
2ac	T	RLQ when delivery acknowledgement is pending and negotiated release is not available	Deliver data as specified in table 9 or 10. Issue VT-RELEASE indication. Sequence ends.
3ab	I	DLQ with request for acknowledgement instead of RLR	None.
4ab	I	RLR with VT-result = "failure"	Issue VT-RELEASE confirm with VT-result = "failure" and failure-reason = "collision-detected". Perform normal delivery action. Issue VT-DELIVER indication with request for acknowledgement. Proceed at step 3 of table 13 as the target VTPM.
Agreed Release without Token			
1b	I	VT-RELEASE, when tokens not held	Send RTQ.
2ba	T	RTQ, holding tokens	Send GTQ.
3baa	I	GTQ	Forward data. Send RLQ.
4baaa	T	RLQ without collision	Perform data delivery as specified in table 9 or 10. Issue VT-RELEASE indication.
5baaaa	T	VT-RELEASE response with VT-result ="success"	Forward data as specified in table 9 or 10. Send RLR with VT-result = "success".
6baaaa	I	RLR with VT-result = "success"	Deliver data as specified in table 9 or 10. Issue VT-RELEASE confirm with VT-result = "success". Sequence ends.

Table 7 (concluded)

Step	VTPM	Event	Action
Refusal of Release			
5baaab	T	VT-RELEASE response with VT-result = "failure"	Send RLR with VT-result = "failure".
6baaab	I	RLR with VT-result = "failure"	Issue VT-RELEASE confirm with VT-result = "failure" (take no action in respect of data in transit). Sequence ends.
Release Collision			
2bb	T	RTQ and tokens not held (because of collision with token passing procedure)	None.
3bb	I	GTQ	Proceed with step 3baa of table 7.
2bc	T	RTQ when awaiting RLR	None (continue to await RLR)
3bab	I	RLQ instead of GTQ	Issue VT-RELEASE confirm with VT-result = "failure" and failure-reason = "collision-detected". Proceed with step 2aa of table 7 as the target VTPM.
4baab	T	RLQ when delivery acknowledgement is pending	Send RLR with VT-result = "failure".
5baab	I	DLQ with request for acknowledgement instead of RLR	None.
6baab	I	RLR with VT-result = "failure"	Issue VT-RELEASE confirm with VT-result = "failure" and failure-reason = "collision-detected". Perform normal delivery action. Issue VT-DELIVER indication with request for acknowledgement. Proceed at step 3 of table 13 as the target VTPM.

Table 8 - Unconditional Release procedures

Step	VTPM	Event	Action
User Initiated Termination			
1	I	VT-U-ABORT request	Send AUQ with VT-user-failure-reason taken from the VT-U-ABORT request (any data in transit or other actions in progress are abandoned).
2	T	AUQ and VT-ASSOCIATE indication has been issued by the VTPM and the VT-User has issued neither a VT-U-ABORT request nor a VT-RELEASE response with VT-result = "success"	Issue a VT-U-ABORT indication with VT-user-failure-reason parameter from AUQ.

Table 8 (concluded)

Step	VTPM	Event	Action
VTPM Initiated Termination			
1	I	Some internal problem which prevents the VTPM from continuing normally	Send APQ with VT-reason = "local-error". Issue a VT-P-ABORT indication with appropriate VT-reason (data in transit or other actions in progress are abandoned).
2	T	APQ with VT-reason = "local-error" and VT-ASSOCIATE indication has been issued by the VTPM and VT-User has issued neither a VT-U-ABORT request nor a VT-RELEASE response with VT-result = "success"	Issue a VT-P-ABORT indication with appropriate VT-reason parameter.
Supporting Service Initiated Termination			
1	E	PAB	Abandon any data in transit and actions in progress. Issue VT-P-ABORT indication with VT-reason information as provided by the supporting service.

Table 9 - S-mode Normal Data Transfer procedures

Step	VTPM	Event	Action
1	S	VT-DATA request addressing a control object with CO-priority = "normal" or a display object. At the convenience of the VTPM	Send NDQ containing the updates in a manner so as to maintain the relative sequence of CCA object updates.
2a	S	VT-DELIVER, VT-GIVE-TOKEN or VT-RELEASE request	Send any NDQs not yet forwarded. Take action appropriate to primitive.
2b	S	VT-DATA request addressing a control object for which CO-trigger = "selected"	Send any NDQs not yet forwarded. Send GTQ.
3aa/3ba	R	NDQ when no-delivery-control or simple-delivery-control is in effect	Process the updates so as to maintain the order of updates. Issue VT-DATA indication maintaining the 1:1 relation of requests and indications.
3ab/3bb	R	DLQ, GTQ, or RLQ	Process the updates so as to maintain the order of updates. Issue VT-DATA indication maintaining the 1:1 relation of requests and indications. Sequence ends.
3ac/3bc	R	NDQ when quarantined delivery control is in effect	Store updates in NDQ (the number of indications may in effect not be 1:1 with the number of VT-DATA requests due to possible net effecting. For the timing of this see events for DLQ, RLQ, and GTQ).

Table 10 - A-mode Normal Data Transfer procedures

Step	VTPM	Event	Action
1	S	VT-DATA request to a control object for which CO-priority = "normal", or a Display Object. At the convenience of the VTPM	Send NDQ expressing the updates and maintaining the relative sequence of display and control object updates. (If quarantine-delivery-control is in effect there need not be a 1:1 correspondence between VT-DATA requests and NDQs (net effecting).)
2a	R	NDQ when no-delivery-control is in effect	Process updates so as to maintain the order of updates. Issue VT-DATA indication maintaining the 1:1 relation of requests and indications.
2b	R	NDQ when simple-delivery-control is in effect	Process updates in the NDQ so as to maintain the order of updates. Issue VT-DATA indication maintaining the 1:1 relation of requests and indications. (The receipt of other PDUs will cause the updates to be applied and will send NDQ actions as well, see DLQ and RLQ.)
2c	R	NDQ when quarantine-delivery-control is in effect	Store updates in NDQ. (The number of indications may not be 1:1 with the number of VT-DATA requests, due to possible net effecting. For the timing of this see events for DLQ.)
3	S	VT-DELIVER or VT-RELEASE request, or VT-RELEASE response with VT-result = "accepted", or a VT-DATA request addressing a control object for which CO-trigger = "selected"	Send NDQs in accordance with any CCA updates received in preceding VT-DATA requests but not yet forwarded by the VTPM, maintaining the relative sequence of display and control object updates. In the case of a "trigger" object, the update to that object is transmitted as the whole or part of a final NDQ.
4	R	RLQ, DLQ or RLR with VT-result = "accepted", or NDQ updating a control object for which CO-trigger = "selected"	Update the local CCA replica with any updates specified in previous NDQs but not yet implemented, maintaining the relative sequence of display and control object updates. Issue VT-DATA indications in accordance with updates to CCA specified in previous NDQs not yet notified to the VT-user and maintaining their relative sequence. (If quarantine-delivery-control is in effect, there is not necessarily a 1:1 correspondence between the VT-DATA indications and the originating VT-DATA requests (due to net effecting). Otherwise there is a 1:1 correspondence. In the case of RLR, RLQ and DLQ protocol data units, further action is specified under release and delivery, see tables 7, 8, & 13.)

Table 11 - Data Transfer for High Priority Objects procedure

Step	VTPM	Event	Action
1	S	VT-DATA request addressing a control object with CO-priority = "high"	As soon as practicable, send HDQ containing update. (Sequence relative to ASR, SPR, or ENR must be maintained but the sequence relative to other operations need not be observed.)
2	R	HDQ	As soon as practicable deliver HDQ data to VT-user and issue a VT-DATA indication primitive corresponding to the HDQ contents.

Table 12 - Data Transfer for Urgent Objects procedures

Step	VTPM	Event	Action
1	S	VT-DATA request addressing a control object with CO-priority = "urgent"	Send UDQ expressing the update. (Sequence relative to SPR or ENR must be maintained but the sequence relative to other operations need not be observed.)
2	R	UDQ	Issue a VT-DATA indication primitive corresponding to the UDQ contents. (The sequence relative to other primitives is not observed.)

Table 13 - Delivery Control procedure

Step	VTPM	Event	Action
1	I	VT-DELIVER request	Forward data as specified in table 9 or 10. Send DLQ
2	T	DLQ	Perform data delivery actions as specified in table 9 or 10. Issue a VT-DELIVER indication. If acknowledgement not requested: sequence ends.
Acknowledgement procedure			
3	T	VT-ACK-RECEIPT request	Send DAQ.
4a/5b/5d	I	DAQ	Issue a VT-ACK-RECEIPT confirm.
4b	I	RLQ instead of DAQ and negotiated release is available	Send RLR with VT-result = "failure" and failure-reason = "collision-detected".
4c	I	RLQ instead of DAQ and negotiated release is not available	Deliver data as specified in table 9 or 10. Issue VT-RELEASE indication. Proceed at step 3aaa or 5baaaa of table 7.
4d	I	SPQ or SNQ instead of DAQ	Send SPR or SNR with VT-result = "failure" and failure-reason = "collision-detected".

Table 14 - Token Management procedures

Step	VTPM	Event	Action
Passing the tokens			
1	I	VT-GIVE-TOKEN request, or a VT-DATA request addressing a control object for which CO-trigger = "selected"	Forward data as specified in table 9. Send GTQ and thereafter perform the actions assigned to the VTPM not holding the tokens.
2	T	GTQ	Perform any delivery actions as specified in tables 9 & 13. Issue a VT-GIVE-TOKEN indication and thereafter perform the actions assigned to the token-holding VTPM.
Requesting the tokens			
1	I	VT-REQUEST-TOKEN request	Send RTQ.
2a	T	RTQ and holding the token	Issue a VT-REQUEST-TOKEN indication.
2b	T	RTQ and not holding token (because of a collision with the token-passing procedure)	Take no action.

Table 15 - Break and Exception procedures

Step	VTPM	Event	Action
1a	I	VT-BREAK request	If EXR is expected, store VT-BREAK parameters. Otherwise set contents of DOs and COs to that of the reset context. Clear out any outstanding wait for acknowledgement. Send BKQ.
2a	T	BKQ	Set contents of DOs and COs to that of the reset context. Clear any outstanding wait for acknowledgement. Clear out any stored VT-BREAK parameters. Issue VT-BREAK indication.
3a	T	VT-BREAK response	Send BKR. Proceed using data transfer procedures.
4a	I	BKR	Issue VT-BREAK confirm. Proceed using data transfer procedures.
1b	I	Exception found	Clear out any outstanding wait for acknowledgement. Send EXQ and issue VT-EXCEPTION indication.
2b	T	EXQ	Clear out any outstanding wait for acknowledgement. Send EXR and issue VT-EXCEPTION indication.
3b	I	EXR	Process stored VT-BREAK if present as in 1a and go to 2a.

9 Switch Profile procedures

9.1 Association Establishment

The establishment procedures when Switch Profile negotiation functional unit is included are similar to the primary procedures, see tables 5 and 6. References to VT-result value "success-with-warning" are added to the events of 3ab and 4ab. Step 3aa of table 16 replaces step 3aa of table 5 and step 4aac of table 16 is added to table 6.

On completion of the establishment procedure, if the ASR PDU contained VT-result value "success", the data handling phase is entered. If, however, the ASR PDU contained VT-result value "success-with-warning", negotiation quiescent phase is entered.

9.2 Agreed Release

The release procedures when Switch Profile negotiation is available are the same as the procedures in 8.2, see table 7, with an additional step which is defined in table 17.

9.3 Unconditional Termination

The termination procedures when Switch Profile negotiation is available are the same as the primary procedures, see table 8.

9.4 Negotiation

The procedures for Switch Profile negotiation are defined in table 18. Switch Profile negotiation may be used at any time during the lifetime of the association; it is not limited to use before data transfer has taken place. If the result of association establishment was "success with warning" this negotiation must be performed before data transfer can take place.

9.5 Data Transfer

The data transfer procedures when Switch Profile negotiation is available are the same as the primary procedures, see Tables 9, 10 and 11, with the addition of

a) VT-SWITCH-PROFILE to the list of request primitives which cause data forwarding (in step 3a of table 9 and step 4 of table 10);

b) SPQ to the list of protocol elements which cause delivery action (in step 5ac/5bc and 6ab/6bb of table 9 and step 5 of table 10);

c) VT-SWITCH-PROFILE to the sequencing requirements in step 4 of table 11.

9.6 Delivery Control

The delivery control procedures when Switch Profile negotiation is available are the same as the procedures in 8.6, see table 13, except that in step 4b "RLQ" is replaced by "RLQ or SPQ".

Table 16 - Association Establishment procedures for Switch Profile

Step	VTPM	Event	Action
3aa (amended)	T	VT-ASSOCIATE response with VT-result = "success" or "success-with-warning"	Send ASR with same VT-result value. If VT-result = "success": establish "reset context" according to the contents of all DOs and COs.
4aac (new step)	I	ASR with VT-result = "success-with-warning"	Issue VT-ASSOCIATE confirm with same VT-result value.

Table 17 - Termination procedures for Switch Profile

Step	VTPM	Event	Action
3bac	I	SPQ instead of GTQ	Issue VT-RELEASE confirm with VT-result = "failure" and VT-reason = "collision-detected". Proceed with step 2 of table 18 as the target VTPM.

Table 18 - Switch Profile procedures

Step	VTPM	Event	Action
1a	I	VT-SWITCH-PROFILE request and holding the token	Forward data as described in table 9 or 10. Send SPQ.
1b	I	VT-SWITCH-PROFILE request and token not held (A-mode)	Forward data as described in table 9 or 10. Send RTQ.
2aa/4baa	T	SPQ with acceptable content	Deliver data as described in table 9 or or 10. Issue VT-SWITCH-PROFILE indication.
2ab/4bab	T	SPQ with unacceptable contents	Send SPR with VT-result = "failure" and the appropriate failure-reason.
2ac	I	RTQ instead of SPR	None (continue to await SPR).
2ad	I	DLQ instead of SPR and acknowledgement requested	Issue VT-SWITCH-PROFILE confirm with VT-result = "failure" and VT-provider-failure-reason = "collision-detected". Deliver data as described in table 9 or 10. Issue VT-DELIVER indication.
2ae	I	DLQ instead of SPR and acknowledgement not requested	Deliver data as described in table 9 or 10. Issue VT-DELIVER indication. (Continue to await SPR).
2ba	T	RTQ (A-mode)	Send GTQ.
2bb	I	RLQ or SPQ instead of GTQ	Issue VT-SWITCH-PROFILE confirm, VT-result = "failure", VT-reason = "collision-detected". Proceed with this procedure, that for Start Negotiation, or that for Release as the target VTPM.
3aa	T	VT-SWITCH-PROFILE response	If A-mode and VT-result = "success" : forward data as described in table 10. If VT-result = "success" : establish "reset-context". Send SPR.
3ba	I	GTQ	Send SPQ.
4aa/3ab	I	SPR	If A-mode and VT-result = "success" : deliver data as described in table 10. If VT-result = "success" : establish "reset-context".

9.7 Token Management

The token management procedures when Switch Profile negotiation is available are the same as the procedures in 8.7, see table 14.

9.8 Break and Exception

The procedures for Switch Profile negotiation are the same as the procedures in 8.8, see table 15.

10 Multiple Interaction Negotiation procedures

10.1 Association Establishment

The establishment procedures when Multiple Interaction Negotiation (MIN) functional unit is included are the same as the procedures in 9.1.

Table 19 - Termination procedures for MIN

Step	VTPM	Event	Action
3bad	I	SNQ instead of GTQ	Issue VT-RELEASE confirm with VT-result = "failure" and VT-provider-failure-reason = "collision-detected". Proceed with step 2 of table 20 as the target VTPM.

10.2 Agreed Release

The release procedures when Multiple Interaction Negotiation is available are the same as the procedures in 9.2, see tables 7 and 17, with an additional step which is defined in table 19.

10.3 Unconditional Termination

The termination procedures when Multiple Interaction negotiation is available are the same as the procedures in 9.3.

10.4 Negotiation

When Switch Profile Negotiation and Multiple Interaction Negotiation are both available, they may be used at any time during the lifetime of the association. They are not limited to use before data transfer has taken place. If the result of association establishment was "success-with-warning" one or the other type of negotiation must be performed before data transfer can take place.

10.4.1 Switch Profile Negotiation

The procedures for Switch Profile negotiation when Multiple Interaction Negotiation is available are the same as in 9.4 except that in the action of step 2bb of table 18 "RLQ or SPQ" is replaced by "RLQ, SPQ or SNQ"

10.4.2 Multiple Interaction Negotiation

The procedures for Multiple Interaction Negotiation are divided into three phases.

10.4.2.1 The establishment of Multiple Interaction Negotiation (i.e., entry into the Negotiation phase of the VT service) is defined in table 20.

In S-mode the initiating VTPM must be that which holds the token. In A-mode either VTPM may initiate the sequence. In A-mode collisions within MIN are resolved in favour of the VTPM designated as the "collision winner".

10.4.2.2 The negotiation of parameter values is defined in table 21. The negotiation of values for each VT parameter is independent of the negotiation of values for all other parameters, but information for more than one parameter may be combined into a single protocol element. More than one protocol element may be returned in response to such a protocol element. (For example NOQ offering values for a number of parameters may result in NAQ accepting values for some parameters and also NJQ rejecting values for others.) The sequences apply on a per parameter basis; multiple instances of these sequences may concurrently be in progress and a single instance may split into multiple instances which then proceed independently.

Table 20 - Establishment of Multiple Interaction Negotiation procedures

These procedures are the same as table 18 with the following changes:

replace:	with:
VT-SWITCH-PROFILE	VT-START-NEGOTIATION
SPQ	SNQ
SPR	SNR

delete references to reset-context in steps 3aa and 4aa/3ab

There are two types of value negotiation sequence, starting with VT-NEG-INVITE and VT-NEG-OFFER respectively.

In S-mode the tokens must be held by the VTPM before it sends a negotiation protocol element. In A-mode only the initiator of the MIN establishment sequence may initiate a value negotiation sequence.

10.4.2.3 The termination of Multiple Interaction Negotiation (i.e., exit from the Negotiation phase of the VT service) is specified in table 22.

The termination of Multiple Interaction Negotiation includes agreement on the retaining or discarding of the negotiated VTE and on entry to the Data Handling phase.

10.5 Data Transfer

Data Transfer procedures are the same as in 8.5 but with the addition of

a) VT-SWITCH-PROFILE and VT-START-NEG to the list of protocol elements which cause data forwarding in step 2a of table 9 and step 3 of table 10;

b) SPQ and SNQ to the list of protocol elements which cause delivery action in step 3ab/3bb of table 9 and step 4 of table 10.

Table 21 - Negotiation of parameter values procedures

Step	VTPM	Event	Action
MIN Initiated by VT-NEG-INVITE			
1a	I	VT-NEG-INVITE request	Send NIQ with the same parameter identities.
2a	T	NIQ	Issue VT-NEG-INVITE indication with the same parameter identities.
3a	T	VT-NEG-OFFER request	Send NOQ with the same parameter values. Proceed at step 4bc.
MIN Initiated by VT-NEG-OFFER			
1b	I	VT-NEG-OFFER request	Send NOQ with the same parameter values.
2b	T	NOQ	Issue VT-NEG-OFFER indication.
3ba	T	VT-NEG-ACCEPT request	Send NAQ with the same parameter values.
4ba	I	NAQ	Issue VT-NEG-ACCEPT indication with the same parameter values. Sequence ends.
3bb	T	VT-NEG-REJECT request	Send NJQ with the same parameter values.
4bb	I	NJQ	Issue VT-NEG-REJECT indication with the same parameter values. Sequence ends.
3bc	T	VT-NEG-OFFER request (signifying a counter offer)	Send NOQ with the same parameter values.
4bc	I	NOQ	Issue VT-NEG-OFFER indication with acceptable parameter values or "reduced-to-null" for all unacceptable parameters.
5bca	I	VT-NEG-ACCEPT request	Send NAQ with the same parameter values.
6bca	T	NAQ	Issue VT-NEG-ACCEPT indication with the same parameter values. Sequence ends.
5bcb	I	VT-NEG-REJECT request	Send NJQ with the same parameter values.
6bcb	T	NJQ	Issue VT-NEG-REJECT indication with the same parameter values. Sequence ends.

10.6 Delivery Control

The procedures are the same as in table 13 except that in step 4b, "RLQ" is replaced by "RLQ, SPQ or SNQ".

10.7 Token Management

The procedures are the same as in table 14.

10.8 Break and Exception

The procedures are the same as in 8.8, see table 15.

11 Mapping of protocol elements

The protocol elements defined in clause 6 are transferred between peer VTPMs using the supporting services. Table 23 specifies the mapping of each protocol element onto an Association Control or Presentation service. Those parameter values required by this part of ISO/IEC 9041 for the primitives of these services are specified below. The provider abort (PAB) event is mapped to an A-P-ABORT indication.

NOTE – Values for those ACSE and Presentation parameters for which no value is specified in this clause are outside the scope of this part of ISO/IEC 9041.

Table 22 - Termination of Multiple Interaction Negotiation procedures

Step	VTPM	Event	Action
1a	I	VT-END-NEG request with VT-vte-choice = "draft" and draft-VTE incomplete	Issue VT-END-NEG confirm with VT-result = "failure", VT-provider-failure-reason = "VTE-incomplete". Sequence ends.
1b	I	VT-END-NEG request with VT-vte-choice = "either" and draft-VTE incomplete	Set VT-vte-choice = "current". Send ENQ.
1c	I	VT-END-NEG request (all other cases)	Send ENQ. Proceed to step 2b.
2b	T	ENQ	Issue VT-END-NEG primitive with parameter as derived from the ENQ PDU.
3b	T	VT-END-NEG response	Send ENR with parameter values from response. If VT-result = "success" and VT-choice = "draft" : set draft-VTE as current-VTE. Establish "reset-context". Sequence ends.

11.1 Mapping to Association Control Services

11.1.1 A-ASSOCIATE request and indication

The A-ASSOCIATE request and indication primitives convey the ASQ protocol element. This part of ISO/IEC 9041 defines the following content for the A-ASSOCIATE parameters:

a) Called and Calling Application Entity Titles convey the corresponding parameters of ASQ;

NOTE – Application Entity Title is used here for the collection of four naming parameters defined in ISO/IEC 8649, namely, application-process-title, application-process-invocation-identifier, application-entity-qualifier and application-entity-invocation-identifier. The relation between these naming concepts is defined in ISO/IEC 7498-3.

b) Application Context Name contains the identifier specified in Annex B;

c) An abstract syntax component of the Presentation Context Definition List contains the identifier specified in Annex B;

d) Session Requirements contains

1) Typed Data functional unit;
2) Duplex functional unit when VT-mode has value "A-mode", otherwise Half-duplex functional unit, corresponding to the VT-mode parameter of ASQ;
3) Resynchronization functional unit if the Break VT functional unit was selected;
4) Expedited Data functional unit if the Urgent Data VT functional unit was selected;
5) Negotiated Release functional unit if the Negotiated Release VT functional unit was selected;
6) Major Synchronization functional unit if either of the negotiation VT functional units or the Break VT functional unit was selected;

e) Initial Synchronization Point Serial Number takes the value 0;

f) For S-mode, Initial Assignment of Tokens parameter takes one of the values

- "requestor-side",
- "acceptor-side",
- "acceptor-chooses",

for all tokens, corresponding to the VT-WAVAR-owner parameter of ASQ;

g) For A-mode, Initial Assignment of Tokens parameter takes the value "requestor-side" for all tokens;

h) User Information parameter contains the ASQ-pdu structured as required by clause 12;

i) P-context Definition List must contain at least the pair vt-b-syntax, {joint-iso-ccitt asn1(1) basic-encoding(1)}, see annex B.

11.1.2 A-ASSOCIATE response and confirm

The A-ASSOCIATE response and confirm primitives convey the ASR protocol element. This part of ISO/IEC 9041 defines the following content for the A-ASSOCIATE parameters:

a) User Information conveys the ASR-pdu structured as required by clause 12;

Table 23 - Protocol element mapping

Protocol Element	Mapping
ASQ	A-ASSOCIATE request and indication
ASR	A-ASSOCIATE response and confirm
RLQ	A-RELEASE request and indication
RLR	A-RELEASE response and confirm
AUQ	A-ABORT request and indication
APQ	A-ABORT request and indication
NDQ	P-DATA request and indication
	P-TYPED-DATA request and indication
HDQ	P-TYPED-DATA request and indication
UDQ	P-EXPEDITED-DATA request and indication
	P-TYPED-DATA request and indication
DLQ	P-DATA request and indication
	P-TYPED-DATA request and indication
DAQ	P-TYPED-DATA request and indication
BKQ	P-RESYNCHRONIZE request and indication
BKR	P-RESYNCHRONIZE response and confirm
EXQ	P-RESYNCHRONIZE request and indication
EXR	P-RESYNCHRONIZE response and confirm
GTQ	P-TOKEN-GIVE request and indication
RTQ	P-TOKEN-PLEASE request and indication
SPQ	P-SYNCHRONIZE-MAJOR request and indication
SPR	P-SYNCHRONIZE-MAJOR response and confirm
SNQ	P-SYNCHRONIZE-MAJOR request and indication
SNR	P-SYNCHRONIZE-MAJOR response and confirm
ENQ	P-SYNCHRONIZE-MAJOR request and indication
ENR	P-SYNCHRONIZE-MAJOR response and confirm
NIQ	P-DATA request and indication
	P-TYPED-DATA request and indication
NOQ	P-DATA request and indication
	P-TYPED-DATA request and indication
NAQ	P-DATA request and indication
	P-TYPED-DATA request and indication
NJQ	P-DATA request and indication
	P-TYPED-DATA request and indication

b) Result conveys the VT-result parameter of ASR with the following mapping:

ASR Parameters	ACSE Parameters
Result="success"	Result="accepted by responder"
Result="success-with-warning"	Result="accepted by responder"
Result="failure"	Result="rejected by responder" and "reason not specified"

c) Responding Application Entity Title conveys the corresponding parameter of ASR;

NOTE – See Note in 11.1.1.

d) Initial Assignment of Tokens corresponds to the VT-WAVAR-Owner parameter of ASR for all defined tokens.

11.1.3 A-RELEASE request and indication

The A-RELEASE request and indication primitives convey the RLQ protocol element. No parameter values for these primitives are defined by this part of ISO/IEC 9041.

11.1.4 A-RELEASE response and confirm

The A-RELEASE response and confirm primitives convey the RLR protocol element. This part of ISO/IEC 9041 defines the following content for the A-RELEASE parameters:

a) User Information conveys the RLR-pdu structured as required by clause 12;

b) Result conveys the VT-result parameter of RLR.

11.1.5 A-ABORT request and indication

The A-ABORT request and indication primitives convey the AUQ and APQ protocol elements. This part of ISO/IEC 9041 defines the following contents for the parameters of these primitives:

a) the Abort Source parameter of A-ABORT indication takes the value "ACSE user";

b) User Information conveys the AUQ-pdu or the APQ-pdu, both structured as required by clause 12.

11.1.6 A-P-ABORT indication

The A-P-ABORT indication primitive conveys the PAB event of the VT protocol.

11.2 Use of Presentation Services

11.2.1 P-DATA request and indication

The P-DATA request and indication primitives convey the following protocol elements:

NAQ, NDQ, NIQ, NJQ, NOQ, DLQ.

This part of ISO/IEC 9041 defines the content of the User Data parameter of these primitives to be the PDUs corresponding to the above elements structured as required by clause 12.

The NAQ, NIQ, NJQ and NOQ pdus may be concatenated with themselves and with each other in any combination in a single P-DATA request and indication primitive. Two or more NDQ pdus may be concatenated and such a concatenation may be concenated with a DLQ pdu provided the DLQ pdu is the last in the concenated sequence.

11.2.2 P-EXPEDITED-DATA request and indication

The P-EXPEDITED-DATA request and indication primitives convey the UDQ protocol element. This part of ISO/IEC 9041 defines the content of the User Data parameter of these primitives to be the PDU corresponding to this protocol element structured as required by clause 12. This is an optional service whose use may not have been negotiated (see P-TYPED-DATA).

11.2.3 P-RESYNCHRONIZE request and indication

The P–RESYNCHRONIZE request and indication primitives convey the BKQ and EXQ protocol elements. This International Standard defines the following content for the P–RESYNCHRONIZE parameters:

a) User data contains the BKQ-pdu or the EXQ-pdu structured as required by clause 12;

b) Resynchronize type contains the value "restart";

c) Synchronization point serial number contains values as defined by ISO/IEC 8326;

d) For A-mode, Tokens has the value "acceptor chooses" for the all defined tokens;

e) For S-mode, Tokens takes values from the initiating VT–BREAK request primitive or for EXQ the value "acceptor chooses".

11.2.4 P-RESYNCHRONIZE response and confirm

The P–RESYNCHRONIZE response and confirm primitives convey the BKR and EXR protocol elements. This International Standard defines the following content for the P–RESYNCHRONIZE parameters:

a) User Data contains the BKR-pdu structured as defined by clause 12; EXR does not require the User Data field;

b) For A-mode, Tokens takes values from the existing token position;

c) For S-mode, Tokens takes its value from the responding VT–BREAK response primitive or for EXR from the existing token position.

11.2.5 P-SYNCHRONIZE-MAJOR request and indication

The P-SYNCHRONIZE-MAJOR request and indication primitives convey the following protocol elements:

ENQ, SNQ, SPQ.

This part of ISO/IEC 9041 defines the content of the User Data parameter of these primitives to be the PDUs corresponding to the above elements structured as required by clause 12.

11.2.6 P-SYNCHRONIZE-MAJOR response and confirm

The P-SYNCHRONIZE-MAJOR response and confirm primitives convey the following protocol elements:

ENR, SNR, SPR.

This part of ISO/IEC 9041 defines the content of the User Data parameter of these primitives to be the PDUs corresponding to the above elements structured as required by clause 12.

11.2.7 P-TOKEN-GIVE request and indication

The P-TOKEN-GIVE request and indication primitives convey the GTQ protocol element. This part of ISO/IEC 9041 defines the content of the Tokens parameter to be all defined tokens.

11.2.8 P-TOKEN-PLEASE request and indication

The P-TOKEN-PLEASE request and indication primitives convey the RTQ protocol element. This part of ISO/IEC 9041 defines the content of the Tokens parameter to be all defined tokens.

11.2.9 P-TYPED-DATA request and indication

The P-TYPED-DATA request and indication primitives convey the following protocol elements:

DAQ, HDQ, UDQ, NDQ, DLQ, NAQ, NIQ, NJQ, NOQ.

The NDQ, DLQ, NAQ, NIQ, NJQ, NOQ elements are only carried by these primitives when operating in A-mode and the session connection is half-duplex.

The UDQ element is only carried by these primitives when the Expedited Data session functional unit was not successfully negotiated.

This part of ISO/IEC 9041 defines the content of the User Data parameter of these primitives to be the PDUs corresponding to the above elements structured as required by clause 12.

The NAQ, NIQ, NJQ and NOQ pdus may be concatenated with themselves and with each other in any combination in a single P-TYPED-DATA request and indication primitive. Two or more NDQ pdus and a DAQ pdu may be concatenated in any order and such a concatenation may be concenated with a DLQ pdu provided the DLQ pdu is the last in the concenated sequence. Two or more HDQ pdus may be concenated. Two or more UDQ pdus may be concenated. NDQ, HDQ and UDQ pdus may not occur in the same concatenation.

12 Protocol data unit structure

This structure is expressed in ASN.1 notation. Where a component part of a structure is stated as OPTIONAL and the significance of presence/absence is not given explicitly, this significance can be deduced from the description of the protocol data unit in an earlier clause or from the VT Service defined in ISO/IEC 9040.

12.1 General format

```
ISO9041-VTP DEFINITIONS ::= BEGIN
BasicVTPitem ::= CHOICE
        { asq-pdu           [0]  IMPLICIT ASQcontent,
          asr-pdu           [1]  IMPLICIT ASRcontent,
          rlr-pdu           [2]  G.Result2,
          auq-pdu           [3]  IMPLICIT PrintableString,
                                 -- absence of a VT-user-failure-reason is represented by the empty string
          apq-pdu           [4]  IMPLICIT PrintableString,
                                 -- empty string if reason not provided
                                 -- "P" if reason = protocol-error
                                 -- "L" if reason = local-error
          hdq-pdu           [5]  IMPLICIT SEQUENCE OF G.COupdate,
          ndq-pdu           [6]  IMPLICIT NDQcontent,
          udq-pdu           [7]  IMPLICIT G.COupdate,

          bkq-pdu           [8]  IMPLICIT BKQcontent,
          bkr-pdu           [9]  IMPLICIT BKQcontent,
          dlq-pdu           [10] IMPLICIT BOOLEAN,
                                 -- true if acknowledge required
          daq-pdu           [11] IMPLICIT NULL,

          spq-pdu           [12] IMPLICIT G.Profile,
          spr-pdu           [13] IMPLICIT SNRcontent,

          snq-pdu           [14] IMPLICIT G.Profile,
          snr-pdu           [15] IMPLICIT SNRcontent,
          enq-pdu           [16] IMPLICIT ENQcontent,
          enr-pdu           [17] IMPLICIT ENRcontent,
          niq-pdu           [18] IMPLICIT G.ParamIdentList,
          noq-pdu           [19] IMPLICIT G.ParamOfferList,
          naq-pdu           [20] IMPLICIT G.ParamValueList,
          njq-pdu           [21] IMPLICIT G.ParamIdentList
          exq-pdu           [22] IMPLICIT EXQcontent
          spq2-pdu          [23] IMPLICIT SPQ2content
        -- This form of spq shall be used when context retention is required. Either form of spq is allowed
        --   when context retention is not required, ~~even~~ when the Context Retention functional unit is selected.
        -- spq2 shall not be used when the Context Retention functional unit is not selected.
          spr2-pdu          [24]  IMPLICIT SPR2content,
        -- This form of spr shall be used when context retention is required. Either form of spr is allowed
        --   when context retention is not required, ~~even~~ when the Context Retention functional unit is selected.
        -- spq2 shall not be used when the Context Retention functional unit is not selected.
         }
        -- In general, each type of PDU thus has its own identifying tag(s) which is/are unique within the protocol;
        --  this is not strictly necessary for those items whose type is uniquely deducible from the
        --  primitives which convey them but is adopted for flexibility and extensibility.
        -- RLQ, GTQ, RTQ and EXR are however identified by the bearer service, in the case of EXR by the absence of
        --  user data in the bearer primitive, see 11.2.4
```

```
ASQcontent ::= SEQUENCE
        { class             [0]  IMPLICIT INTEGER { basic    (1) },
                            [1]  IMPLICIT G.ImplementationIdent OPTIONAL,
                            [2]  IMPLICIT G.FunctionalUnits OPTIONAL,
                            [3]  IMPLICIT G.Profile OPTIONAL,
                            [4]  IMPLICIT G.ProtocolVersion DEFAULT G.version1,
          either            [5]  IMPLICIT INTEGER { a-mode  (0),
                                                    s-mode  (1) } OPTIONAL
                                       -- only present if VT-mode parameter has value "either-A" or "either-S".
          }
        -- The occurrence of other distinctively tagged types in ASQcontent as received by the VTPM is not an error.
----------------------------------------------------------------------------------------------------

ASRcontent ::= SEQUENCE
        {                           G.Result3,
                            [3]  IMPLICIT G.ImplementationIdent      OPTIONAL,
                            [4]  IMPLICIT G.ProtocolVersion          DEFAULT G.version1,
                            [5]  IMPLICIT G.ProfileArgumValueList    OPTIONAL,
                            [6]  IMPLICIT G.FunctionalUnits          OPTIONAL }
        -- The occurrence of other distinctively tagged types in ASRcontent as received by the VTPM is not an error.
----------------------------------------------------------------------------------------------------

BKQcontent ::= SEQUENCE
        { standard          [0]  IMPLICIT SEQUENCE
                                 { pointer              [0]  IMPLICIT G.ExplicitPointer OPTIONAL,
                                   logPointer           [1]  IMPLICIT G.LogExpPointer OPTIONAL
                                 } OPTIONAL,
          profile           [1]  IMPLICIT SEQUENCE OF SEQUENCE
                                                    { ptag INTEGER, OCTET STRING } OPTIONAL,
          stuser            [2]  IMPLICIT SEQUENCE OF SEQUENCE
                                                    { utag INTEGER, OCTET STRING } OPTIONAL }
----------------------------------------------------------------------------------------------------

ENQcontent ::= SEQUENCE {  vteChoice     [0]  IMPLICIT INTEGER { draft    (0),
                                                                 current  (1),
                                                                 either   (2) },
                           failAllowed   [1]  IMPLICIT BOOLEAN OPTIONAL
                                                    -- true = "yes", false or absent = "no"
                           retList       [2]  IMPLICIT SEQUENCE OF PrintableString OPTIONAL }
----------------------------------------------------------------------------------------------------

ENRcontent ::= SEQUENCE {  result              G.Result3,
                           vteChoice     [3]  IMPLICIT BOOLEAN OPTIONAL
                                                    -- true = "draft", false = "current"
                           retList       [2]  IMPLICIT SEQUENCE OF PrintableString OPTIONAL }
----------------------------------------------------------------------------------------------------

SNRcontent ::= SEQUENCE
        {                           G.Result2,
                            [2]  IMPLICIT G.ProfileArgumValueList OPTIONAL }
----------------------------------------------------------------------------------------------------

SPQ2content ::= SEQUENCE { profile        [0]  IMPLICIT G.Profile,
                           retList        [1]  IMPLICIT SEQUENCE OF PrintableString OPTIONAL}
        -- This form of spq shall be used when context retention is required. Either form of spq is allowed when
        --  context retention is not required, even when the Context Retention functional unit is selected.
        -- spq2 shall not be used when the Context Retention functional unit is not selected.
----------------------------------------------------------------------------------------------------
```

```
SPR2content ::= SEQUENCE { result          G.Result2,
                           argList    [2]  IMPLICIT G.ProfileArgumValueList OPTIONAL,
                           retList    [3]  IMPLICIT SEQUENCE OF PrintableString OPTIONAL }
    -- This form of spr shall be used when context retention is required. Either form of spr is allowed when
    --  context retention is not required, even when the Context Retention functional unit is selected.
    -- spq2 shall not be used when the Context Retention functional unit is not selected.
-----------------------------------------------------------------------------------------------------------

EXQcontent ::= SEQUENCE {
      exceptionType    [0]  CHOICE {
                            stdException        [0] IMPLICIT INTEGER  {  rioFull  (0),
                                                                         tooManyFields(1),
                                                                         tooManyFieldElements(2),
                                                                         tooManyFERs(3),
                                                                         tooManyFEIs(4) },
                            proException        [1] IMPLICIT INTEGER },
      standard         [1]  IMPLICIT SEQUENCE {
                            pointer      [0]  IMPLICIT G.ExplicitPointer OPTIONAL,
                            logPointer   [1]  IMPLICIT G.LogExpPointer OPTIONAL } OPTIONAL,
      profile          [2]   IMPLICIT SEQUENCE OF SEQUENCE
                                               {ptag   INTEGER, OCTET STRING  } OPTIONAL }
    -- the service parameter VT-exception-source shall take the value "local" for the sender of the EXQ
    --  and "remote" for the receiver of the EXQ.
-----------------------------------------------------------------------------------------------------------

NDQcontent ::= SEQUENCE {  updates      [0]  IMPLICIT SEQUENCE OF ObjectUpdate,
                           echoNow      [1]  IMPLICIT NULL OPTIONAL,
                                                 -- absence means do not "echo now".
                           startEntry   [2]  IMPLICIT NULL OPTIONAL }
                                                 -- absence means do not "start entry".

ObjectUpdate ::= CHOICE    {  display      [0]  IMPLICIT SEQUENCE
                                                  {  doName     PrintableString OPTIONAL,
                                                     updates    SEQUENCE OF DOupdate },
                              control      [1]  IMPLICIT G.COupdate,
                              rioref       [2]  IMPLICIT G.RIOreference }

DOupdate ::= CHOICE
      { nextXarray       [0]  IMPLICIT NULL,
        nextYarray       [1]  IMPLICIT NULL,
        ptr-relative     [2]  IMPLICIT G.ExplicitPointer,
              -- the values of the service arguments p,q,r (see ISO/IEC 9040) correspond to the items
              -- identified in G.ExplicitPointer as x,y,z respectively
        ptr-absolute     [3]  Pointer,
        text             [4]  IMPLICIT OCTET STRING,
              -- text may incorporate a number of encoded characters; each such character represents
              --  an individual "text" operation as defined in ISO/IEC 9040. The repertoire defines the
              --  encoding of characters into octets. A character may be encoded into one or more octets.
        repeatText       [5]  IMPLICIT SEQUENCE     {     finishAddress    Pointer,
                                                                             OCTET STRING },
        writeAttr        [6]  IMPLICIT SEQUENCE    { AttrId, AttrExtent },
                              -- AttrId encodes both the service arguments attribute-id and attribute-value
        erase            [7]  IMPLICIT SEQUENCE     {     startErase       Pointer,
                                                          endErase         Pointer,
                                                                           EraseAttr },
        previousXarray   [8]  IMPLICIT  NULL,
        previousYarray   [9]  IMPLICIT  NULL,
```

```
nextBlock       [10]  IMPLICIT  NULL,
previousBlock   [11]  IMPLICIT  NULL,
nextField       [12]  IMPLICIT  NULL,
previousField   [13]  IMPLICIT  NULL,
log-relative    [14]  IMPLICIT  G.LogExpPointer,
log-absolute    [15]  LogPointer,
logText         [16]  IMPLICIT SEQUENCE
                          {  fdrAttr          [0] IMPLICIT  BOOLEAN,
                                                  --true = "yes", false = "no"
                             prAttrVal        [1] IMPLICIT  OCTET STRING },
                                -- see comment under text in DOupdate
repeatLogText   [17]  IMPLICIT SEQUENCE
                          {  finishAddress        LogPointer,
                             fdrAttr          [8] IMPLICIT  BOOLEAN,
                                                  -- true = "yes", false = "no"
                             prAttrValStr     [9] IMPLICIT OCTET STRING },
writeLogAttr    [18]  IMPLICIT SEQUENCE     {     AttrId, LogAttrExtent },
                      -- AttrId encodes both the service arguments attribute-id and attribute-value
logErase        [19]  IMPLICIT SEQUENCE
                          {  logStartErase        LogPointer,
                             logEndErase          LogPointer,
                                                  EraseAttr },
createBlock     [20]  IMPLICIT SEQUENCE
                          {  blockPosition [0] IMPLICIT  G.Block,
                             origin        [1] IMPLICIT  G.MeasurePair,
                             dimension     [2] IMPLICIT  G.MeasurePair },
deleteBlock     [21]  IMPLICIT  G.Block
insertXarray    [22]  IMPLICIT INTEGER,
deleteXarray    [23]  IMPLICIT INTEGER,
insertYarray    [24]  IMPLICIT INTEGER,
deleteYarray    [25]  IMPLICIT INTEGER,
copyToBuffer    [26]  SEQUENCE
                      {     address          Pointer,
                            rioName          [10]  IMPLICIT PrintableString OPTIONAL,
                            recordId         [11]  IMPLICIT PrintableString OPTIONAL,
    -- when buffer-name is "temporary", rioName and recordID shall be absent;
    -- when there is only one RIO present in the VTE, rioName is optional, but recordId shall be present
                            rendition        [12]  IMPLICIT NULL OPTIONAL,
                                  -- presence implies "copy attributes", absence implies "no attribute copy"
                            structure        [13]  IMPLICIT INTEGER
                                                   {    none    (0),
                                                        x       (1),
                                                        xAndy   (2)  } OPTIONAL},
                                                        -- absence implies "none"
copyFromBuffer  [27]   SEQUENCE
                      {     address          Pointer,
                            rioName          [10]  IMPLICIT PrintableString OPTIONAL,
                            recordId         [11]  IMPLICIT PrintableString OPTIONAL,
    -- when buffer-name is "temporary", rioName and recordID shall be absent;
    -- when there is only one RIO present in the VTE, rioName is optional, but recordId shall be present
                            rendition        [12]  IMPLICIT NULL OPTIONAL,
                                  -- presence implies "copy attributes", absence implies "no attribute copy"
```

```
                                structure       [13] IMPLICIT INTEGER
                                                     {  none    (0),
                                                        x       (1),
                                                        xAndy   (2)} OPTIONAL,
                                                -- absence implies "none"
                                ripple          [14] IMPLICIT NULL OPTIONAL },
                                                -- presence implies "on", absence implies "off"
        copyLogToBuffer [28]    SEQUENCE
                                {       address             LogPointer,
                                        rioName         [8] IMPLICIT PrintableString OPTIONAL,
                                        recordId        [9] IMPLICIT PrintableString OPTIONAL,
                -- when buffer-name is "temporary", rioName and recordID shall be absent;
                -- when there is only one RIO present in the VTE, rioName is optional, but recordId shall be present
                                        rendition       [10] IMPLICIT NULL OPTIONAL,
                                                -- presence implies "copy attributes", absence implies "no attribute copy"
                                        structure       [11] IMPLICIT NULL OPTIONAL },
                                                -- presence implies "x", absence implies "none"
        copyLogFromBuffer [29]  SEQUENCE
                                {       address             LogPointer,
                                        rioName         [8] IMPLICIT PrintableString OPTIONAL,
                                        recordId        [9] IMPLICIT PrintableString OPTIONAL,
                -- when buffer-name is "temporary", rioName and recordID shall be absent;
                -- when there is only one RIO present in the VTE, rioName is optional, but recordId shall be present
                                        rendition       [10] IMPLICIT NULL OPTIONAL,
                                                -- presence implies "copy attributes", absence implies "no attribute copy"
                                        structure       [11] IMPLICIT NULL OPTIONAL,
                                                -- presence implies "x", absence implies "none"
                                        ripple          [12] IMPLICIT NULL OPTIONAL }
                                                -- presence implies "on", absence implies "off"
    }

Pointer ::= CHOICE
    { current       [0]  IMPLICIT NULL,
      start         [1]  IMPLICIT NULL,
      startX        [2]  IMPLICIT NULL,
      startY        [3]  IMPLICIT NULL,
      end           [4]  IMPLICIT NULL,
      endX          [5]  IMPLICIT NULL,
      endY          [6]  IMPLICIT NULL,
      coords        [7]  IMPLICIT G.ExplicitPointer,
      startB        [8]  IMPLICIT NULL,
      endB          [9]  IMPLICIT NULL }

EraseAttr ::= BOOLEAN
                -- True = erase secondary attributes
```

```
AttrId ::= CHOICE
        { graphicCharacterRepertoire        [0]  IMPLICIT  INTEGER  { null(0) },
          foregroundColour                  [1]  IMPLICIT  INTEGER  { null(0) },
          backgroundColour                  [2]  IMPLICIT  INTEGER  { null(0) },
          emphasis                          [3]  IMPLICIT  PrintableString,
          font                              [4]  IMPLICIT  INTEGER  { null(0) },
                -- Values for the INTEGERs other than zero identify a position in the appropriate list of
                --   assignment VTE-parameters and signify the value of the parameter in that position;
                --   value 1 identifies the first parameter in the list.
          fevGrCharRep                      [5]  IMPLICIT  NULL,
          fevForCol                         [6]  IMPLICIT  NULL,
          fevBakCol                         [7]  IMPLICIT  NULL,
          fevEmph                           [8]  IMPLICIT  NULL,
          fevFont                           [9]  IMPLICIT  NULL  }
                -- Tags 5-9 imply the attribute-value "field-explicit-value"

AttrExtent ::= CHOICE       {  global       [0]  IMPLICIT NULL,
                               address      [1]  IMPLICIT SEQUENCE
                                                    {  beginning   Pointer,
                                                       ending      Pointer },
                               modal        [2]  IMPLICIT  NULL  }

LogPointer ::= CHOICE
        { logCurrent        [0]  IMPLICIT  NULL,
          logStart          [1]  IMPLICIT  NULL,
          logStartF         [2]  IMPLICIT  NULL,
          logStartK         [3]  IMPLICIT  NULL,
          logEnd            [4]  IMPLICIT  NULL,
          logEndF           [5]  IMPLICIT  NULL,
          logEndK           [6]  IMPLICIT  NULL,
          logCoords         [7]  IMPLICIT  G.LogExpPointer  }

LogAttrExtent ::= CHOICE
        { global            [0]  IMPLICIT  NULL,
          address           [1]  IMPLICIT  SEQUENCE
                                   { beginning          LogPointer,
                                     ending             LogPointer  },
          modal             [2]  IMPLICIT NULL  }

END -- of ISO9041-VTP definitions
```

12.2 General Definitions

```
G DEFINITIONS ::= BEGIN
                --* ********************************************** *
                --*          General definitions used at multiple places in the rest of    *
                --*          the syntax but not specific to a particular object            *
                --* ********************************************** *
Block ::= SEQUENCE
        { zValue            [0]  IMPLICIT  INTEGER  OPTIONAL,
          bValue            [1]  IMPLICIT  INTEGER }
-------------------------------------------------------------------------------------------------

COupdate ::= SEQUENCE
        { coName PrintableString,
          objectUpdate  CHOICE
                { characterUpdate    [0]   IMPLICIT  OCTET STRING,
                  booleanUpdate      [1]   IMPLICIT  SEQUENCE
                                           { values      [0]  IMPLICIT BIT STRING,
                                             mask        [1]  IMPLICIT BIT STRING OPTIONAL },
                                  -- If mask is omitted, a bit string with the same length as "values" and a content of
                                  --  all ones is assumed. When the mask is present a "one" bit indicates that the
                                  --  corresponding bit in value is to be used. The new value of the boolean is given by
                                  --        (old value AND (NOT mask) OR (value AND mask)).
                  symbolicUpdate     [2]   IMPLICIT  INTEGER,
                  integerUpdate      [3]   IMPLICIT  INTEGER,
                  bitStringUpdate    [4]   IMPLICIT  BIT  STRING,
                  multiElement       [5]   IMPLICIT  SEQUENCE  OF  SEQUENCE
                      {   identifier         INTEGER,
                          update             CHOICE
                              { characterUpdate    [0]   IMPLICIT  OCTET STRING,
                                booleanUpdate      [1]   IMPLICIT  SEQUENCE
                                                         { values    [0]  IMPLICIT  BIT  STRING
                                                           mask      [1]  IMPLICIT  BIT  STRING  OPTIONAL },
                                                      -- See note under mask in G.COUpdate
                                symbolicUpdate     [2]   IMPLICIT  INTEGER,
                                integerUpdate      [3]   IMPLICIT  INTEGER,
                                bitStringUpdate    [4]   IMPLICIT  BIT  STRING } },

                  cco                [6]   IMPLICIT  CCOupdate,
                  fdco               [7]   IMPLICIT  FDCOupdate,
                  feico              [8]   IMPLICIT  FEICOupdate,
                  fepco              [9]   IMPLICIT  FEPCOupdate,
                  rio               [10]   IMPLICIT  RIOupdate,
                  other             [11]   ANY } }
        -- The choice depends on the value of the VTE-parameter CO-structure as follows:
        --   a)  if CO-structure = 1, then the datatype of COUpdate
        --       is restricted to those defined above corresponding to the tags [0] to [4].
        --   b)  if CO-structure > 1, then
        --       1)  if CO-type-identifier=vt-b-sco-cco, the tag must be [6], see 12.2.1;
        --       2)  otherwise the tag must be [5].
        --   c)  if CO-structure has the value "non-parametric", then the datatype definition is referenced by
        --       the value of the VTE-parameter CO-type-identifier:
        --       1)  vt-b-sco-fdco      – see 12.2.2
        --       2)  vt-b-sco-nullrio   – see 12.2.5
```

```
--          3)  a value for a registered type definition – the CO-type may be found in the register entry.
--              For FEICOs, FEPCOs and RIOs, the data type definition may be found as follows:
--                  i)   feico    – see 12.2.3
--                  ii)  fepco    – see 12.2.4
--                  iii) rio      – see 12.2.5
--          4)  a value for a private type definition – the data type definition is defined entirely
--              by agreement outside the scope of this part of ISO/IEC 9041.
-----------------------------------------------------------------------------------------------------

ExplicitPointer ::= SEQUENCE
        {       x           [0]  IMPLICIT INTEGER OPTIONAL,
                y           [1]  IMPLICIT INTEGER OPTIONAL,
                z           [2]  IMPLICIT INTEGER OPTIONAL,
                b           [3]  IMPLICIT INTEGER OPTIONAL }
                -- When the display object has 2 dimensions then the z value is always omitted.
                -- When the display object has 1 dimension then only the x value is present.
                -- When the block capability is not selected then the b value is always omitted.
                -- Any or all of these values may be omitted as permitted by the operation definitions in ISO/IEC 9040.
-----------------------------------------------------------------------------------------------------

FunctionalUnits ::= BIT STRING
                        {  profileSwitch          (0),
                           multipleIntNeg         (1),
                           negotiatedRelease      (2),
                           urgentData             (3),
                           destructiveBreak       (4),    -- for any bit, value 1 implies offered/accepted,
                           enhancedAccess         (5),    --      value 0 implies not offered/not accepted
                           structuredCOs          (6),    --      depending on occurrence in ASQ or ASR
                           blocks                 (7),
                           fields                 (8),
                           referenceInfOs         (9),
                           ripple                 (10),
                           exceptions             (11),
                           contextRetention       (12) }
-----------------------------------------------------------------------------------------------------

ImplementationIdent ::= SEQUENCE
        { implementationIdentifier      [0]  IMPLICIT OBJECT IDENTIFIER OPTIONAL,
          implementationName            [1]  IMPLICIT PrintableString OPTIONAL,
          implementationVersion         [2]  IMPLICIT PrintableString OPTIONAL }
          -- Intended to allow implementors to identify their implementation in a character format outside
          --      the scope of this part of ISO/IEC 9041.
-----------------------------------------------------------------------------------------------------

ProtocolVersion ::= BIT STRING { version1 (0) }
        -- Further bits may be defined by later editions of this part of ISO/IEC 9041 or in responses to defect reports relating to it.
-----------------------------------------------------------------------------------------------------

IntegerOffer ::= SEQUENCE OF CHOICE
        { individualValue   [0]  IMPLICIT INTEGER,
          range             [1]  IMPLICIT SEQUENCE        {   minimum   INTEGER,
                                                              maximum   INTEGER } }
-----------------------------------------------------------------------------------------------------

LogExpPointer ::= SEQUENCE
        { kValue            [0]  IMPLICIT INTEGER OPTIONAL,
          fValue            [1]  IMPLICIT INTEGER OPTIONAL,
          zValue            [2]  IMPLICIT INTEGER OPTIONAL }
-----------------------------------------------------------------------------------------------------

MeasurePair ::= SEQUENCE
        { xValue            [0]  IMPLICIT INTEGER,
          yValue            [1]  IMPLICIT INTEGER OPTIONAL }
-----------------------------------------------------------------------------------------------------
```

```
ParamIdentList ::= SEQUENCE
        { displayObjects    [0]  IMPLICIT CDS.Identifier OPTIONAL,
          controlObjects    [1]  IMPLICIT CSS.Identifier OPTIONAL,
          deviceObjects     [2]  IMPLICIT DEV.Identifier OPTIONAL,
          deliveryControl   [3]  IMPLICIT NULL OPTIONAL }
----------------------------------------------------------------------------------------------------

ParamOfferList ::= SEQUENCE
        { displayObjects    [0]  IMPLICIT CDS.Offer OPTIONAL,
          controlObjects    [1]  IMPLICIT CSS.Offer OPTIONAL,
          deviceObjects     [2]  IMPLICIT DEV.Offer OPTIONAL,
          deliveryControl   [3]  IMPLICIT BIT STRING    { none        (0),   -- value 1 for a bit implies offer,
                                                          simple      (1),   -- value 0 implies no offer.
                                                          quarantine  (2)    } OPTIONAL }
----------------------------------------------------------------------------------------------------

ParamValueList ::= SEQUENCE
        { displayObjects    [0]  IMPLICIT CDS.Values OPTIONAL,
          controlObjects    [1]  IMPLICIT CSS.Values OPTIONAL,
          deviceObjects     [2]  IMPLICIT DEV.Values OPTIONAL,
          deliveryControl   [3]  IMPLICIT INTEGER       { none        (0),   -- value 1 for a bit implies offer,
                                                          simple      (1),   -- value 0 implies no offer.
                                                          quarantine  (2) } OPTIONAL }
----------------------------------------------------------------------------------------------------

Profile ::= SEQUENCE
        { name              OBJECT IDENTIFIER OPTIONAL
                            ProfileArgumOfferList OPTIONAL }
                -- omission of the name parameter implies the default profile.

ProfileArgumOfferList ::= SEQUENCE
        { specialProfileArgums  [0]  IMPLICIT SEQUENCE OF SEQUENCE
                { identifier            INTEGER,
                  offeredValues         CHOICE
                        { boolean       [0]  IMPLICIT BIT STRING { false (0), true (1) },
                                                -- value 1 for a bit implies offer, 0 implies no offer
                          integer       [1]  IMPLICIT IntegerOffer,
                          string        [2]  IMPLICIT SET OF PrintableString,
                          objid         [3]  IMPLICIT SET OF OBJECT IDENTIFIER }
                } OPTIONAL,
          vteParams             [1]  IMPLICIT ParamOfferList OPTIONAL }
----------------------------------------------------------------------------------------------------

ProfileArgumValueList ::= SEQUENCE
        { specialProfileArgums  [0]  IMPLICIT SEQUENCE OF SEQUENCE
                { identifier  INTEGER,
                  value       CHOICE
                        { BOOLEAN,
                          INTEGER,
                          PrintableString,
                          OBJECT IDENTIFIER } } OPTIONAL,
          vteParams             [1]  IMPLICIT ParamValueList OPTIONAL }
----------------------------------------------------------------------------------------------------

Result3 ::= CHOICE {    success                 [0] IMPLICIT NULL,
                        fail                    [1] IMPLICIT Reason,
                        successWithWarning      [2] IMPLICIT Reason }
```

```
Result2 ::= CHOICE { success < Result3, fail < Result3 }

Reason ::= SEQUENCE {   userA                [0] IMPLICIT PrintableString OPTIONAL,
                        userB                [1] IMPLICIT ErrorCode OPTIONAL,
                        provider             [2] IMPLICIT ErrorCode OPTIONAL }

ErrorCode ::= INTEGER { collisionDetected                (0),
                        vteParamNotSupported             (1),
                        vteParamCombNotSupported         (2),
                        vteIncomplete                    (3),
                        vteProfileNotSupported           (4),
                        vtModeNotSupported               (5),
                        lengthExceeded                   (6)  }
```

```
RIOreference ::= SEQUENCE
    { rioName       [0]  IMPLICIT PrintableString,
                    [1]  IMPLICIT SEQUENCE OF SEQUENCE
                         { recordId      [0]  IMPLICIT PrintableString,
                           operation     [1]  IMPLICIT INTEGER {  executeRecord (0),
                                                                  callRecord (1) } } }

        --*************************************************** *
        --*          The following define types for updates to control objects     *
        --*          whose type and structure are defined in ISO/IEC 9040          *
        --*************************************************** *
```

12.2.1 Context Control Object update syntax

```
CCOupdate ::= SEQUENCE
    { kCoordinate       [0] IMPLICIT INTEGER,
      fCoordinate       [1] IMPLICIT INTEGER,
      zCoordinate       [2] IMPLICIT INTEGER OPTIONAL,
            -- required when VT-dimensions="three"
      entryControlIndex [3] IMPLICIT INTEGER OPTIONAL,
      deviceObjectIndex [4] IMPLICIT INTEGER OPTIONAL,
      feprIndex         [5] IMPLICIT INTEGER OPTIONAL
            -- items with tags [3], [4] and [5] are required on termination update, see ISO/IEC 9040.
      }
    -- This syntax definition is included separately from the multi-element CO update syntax
    --   since the CCO is not partially updatable.
```

12.2.2 Field Definition Control Object update syntax

```
FDCOupdate ::= SEQUENCE OF SEQUENCE
    { labelFCoordinate  [0] IMPLICIT INTEGER,
      labelZCoordinate  [1] IMPLICIT INTEGER OPTIONAL,
      status            [2] IMPLICIT INTEGER { active      (0),
                                               inactive    (1),
                                               void        (2) } OPTIONAL,
      extent            [3] IMPLICIT SEQUENCE OF SEQUENCE
                        { position      [0] IMPLICIT MeasurePair,
                          dimension     [1] IMPLICIT MeasurePair } OPTIONAL,
```

```
    attributes          [4] IMPLICIT SEQUENCE
                          { graphicCharacterRepertoire   [0] IMPLICIT INTEGER OPTIONAL,
                            foregroundColour             [1] IMPLICIT INTEGER OPTIONAL,
                            backgroundColour             [2] IMPLICIT INTEGER OPTIONAL,
                            emphasis                     [3] IMPLICIT PrintableString OPTIONAL,
                            font                         [4] IMPLICIT INTEGER OPTIONAL
                            -- value of zero for any of the integer items in attributes implies the "null" value,
                                                         } OPTIONAL,
    nextField           [5] IMPLICIT INTEGER OPTIONAL,
    previousField       [6] IMPLICIT INTEGER OPTIONAL,
          -- for tags 5 & 6, a zero value implies end of navigation path; a negative value implies "void"
    transmissionPolicy [7] IMPLICIT INTEGER
                          { all                        (0),
                            modifiedAllContent         (1),
                            modifiedPart               (2),
                            none                       (3),
                            refTPCO                    (4) }  OPTIONAL,
    entryControlList    [8] IMPLICIT SEQUENCE OF EntryControl OPTIONAL
    }
  -- If tag 8 is absent, no change is to be made to the entry-control-list in the FDR;
  -- if tag 8 is present but the item has value { } (i.e., is empty), the entry control-list for this FDR is discarded.

EntryControl ::= SEQUENCE
      { deviceObjectList    [0] IMPLICIT SEQUENCE OF PrintableString OPTIONAL,
        feirList            [1] IMPLICIT SEQUENCE OF SEQUENCE
                                { feicoName            PrintableString,
                                  recordIndex          INTEGER }   OPTIONAL,
        feprList            [2] IMPLICIT SEQUENCE OF SEQUENCE
                                { fepcoName            PrintableString,
                                  recordIndex          INTEGER }   OPTIONAL }
  -- If tag 8 of FDCOupdate is present and the item is not empty, each instance of EntryControl is used to update
  --  the corresponding entry-control in the FDR as follows:
  --    – If the instance of EntryControl is empty, no action is taken.
  --    – Otherwise, each part is used to update the corresponding entry-control part as follows:
  --          – If any of the 3 parts is not present, then the corresponding part is not updated;
  --          – If a part is present but empty, the corresponding part is discarded;
  --          – If a part is present and not empty, the list it contains completely replaces the corresponding part.
  --              There is no provision for partial update of such a list.
```

12.2.3 Field Entry Instruction Control Object update syntax

```
FEICOupdate ::= SEQUENCE OF SEQUENCE
                          {   index                [0] IMPLICIT INTEGER,
                              content              [1] IMPLICIT SET OF FEI }

FEI ::= ANY
      -- The syntax for FEI will be found in the register entry pointed
      --  to by the VTE-parameter CO-type-identifier for this FEICO.
```

12.2.4 Field Entry Pilot Control Object update syntax

```
FEPCOupdate ::= SEQUENCE OF SEQUENCE
                    {   index          [0] IMPLICIT INTEGER,
                        event          [1] FEE,
                        condition      [2] IMPLICIT SET OF FEC,
                        reactions      [3] IMPLICIT SEQUENCE OF FER }

FEE ::= ANY             -- field entry event
FEC ::= ANY             -- field entry condition
FER ::= ANY             -- field entry reaction

        -- The syntax for FEI, FEC & FER will be found in the register entry pointed to by
        --  the VTE-parameter CO-Type-Identifier for this FEPCO.
```

12.2.5 RIO Control Object update syntax

```
RIOupdate ::= SEQUENCE OF SEQUENCE
                    {   recordId       [0] IMPLICIT PrintableString OPTIONAL,
                        -- the record id may be omitted if the whole rio is to be updated.
                        operation      [1] IMPLICIT INTEGER  {  eraseRIO      (0),
                                                                deleteRecord  (1),
                                                                createRecord  (2) },
                        updates        [2] IMPLICIT SEQUENCE OF
                                                    ISO9041-VTP.ObjectUpdate OPTIONAL }
        -- "updates" is only present in the case of a Create Record operation.
        -- absence of "updates" for a Create Record operation implies
        --    that the specified record shall be set to the "empty" state.

END -- of G definitions
```

12.3 Conceptual Data Store definitions

```
CDS DEFINITIONS ::= BEGIN
```

12.3.1 CDS Identifiers

```
--* ********************************************** *
--*          The following are to identify parameters to be negotiated     *
--* ********************************************** *

Identifier ::= SET OF SEQUENCE    {  name       PrintableString,
                                     ParameterIdents }
        -- In the ParameterIdents type and its components the occurrence of a null type
        --  indicates that a value for the associated parameter is specified.

ParameterIdents ::= SEQUENCE
    { dimensions      [0]  IMPLICIT  NULL  OPTIONAL,
      xParam          [1]  IMPLICIT  DimensionParamIdent  OPTIONAL,
      yParam          [2]  IMPLICIT  DimensionParamIdent  OPTIONAL,
      zParam          [3]  IMPLICIT  DimensionParamIdent  OPTIONAL,
      erasure         [4]  IMPLICIT  NULL  OPTIONAL,
      repertoire      [5]  IMPLICIT  CompoundRepertoireIdent  OPTIONAL,
      emphasis        [6]  IMPLICIT  CompoundEmphasisIdent  OPTIONAL,
      foreground      [7]  IMPLICIT  CompoundColourIdent  OPTIONAL,
      background      [8]  IMPLICIT  CompoundColourIdent  OPTIONAL,
      access          [9]  IMPLICIT  NULL  OPTIONAL,
      blockParams     [10] IMPLICIT  BlockParamIdent  OPTIONAL,
      fieldParams     [11] IMPLICIT  FieldParamIdent  OPTIONAL,
      rippleCapability [12] IMPLICIT  NULL  OPTIONAL }

DimensionParamIdent ::= SEQUENCE
    { bound           [0]  IMPLICIT NULL OPTIONAL,
      addressing      [1]  IMPLICIT NULL OPTIONAL,
      absolute        [2]  IMPLICIT NULL OPTIONAL,
      window          [3]  IMPLICIT NULL OPTIONAL }

CompoundRepertoireIdent ::= SEQUENCE
    { capability      [0]  IMPLICIT  NULL  OPTIONAL,
                      [1]  IMPLICIT  SEQUENCE  OF  RepertoireFontIdent  OPTIONAL }

RepertoireFontIdent ::= CHOICE
    { NULL,
      SEQUENCE    {   assignment      [0]  IMPLICIT  NULL  OPTIONAL,
                      fontCapability  [1]  IMPLICIT  NULL  OPTIONAL,
                      fontNames       [2]  IMPLICIT  SEQUENCE  OF
                                             AssignmentIdent OPTIONAL } }

AssignmentIdent ::= CHOICE { NULL, PrintableString }
        -- The NULL type in any assignment name position indicates no invitation for values for that assignment;
        --  any value of PrintableString implies an offer for that assignment.
        -- Conventionally the string value should be empty.

CompoundEmphasisIdent ::= SEQUENCE OF AssignmentIdent
```

```
CompoundColourIdent ::= SEQUENCE
                        {   capability          NULL OPTIONAL,
                            assignment          SEQUENCE OF AssignmentIdent OPTIONAL }

BlockParamIdent ::= SEQUENCE
      { capability          [0] IMPLICIT NULL OPTIONAL,
        bound               [1] IMPLICIT NULL OPTIONAL }

FieldParamIdent ::= SEQUENCE
      { capability          [0] IMPLICIT NULL OPTIONAL,
        maxFields           [1] IMPLICIT NULL OPTIONAL,
        maxFieldElements    [2] IMPLICIT NULL OPTIONAL,
        accessOutside       [3] IMPLICIT NULL OPTIONAL }
```

12.3.2 CDS Offers

```
                --****************************************************
                --*          The following are for offered values during negotiation      *
                --****************************************************
Offer ::= SET OF SEQUENCE     {   name          PrintableString,
                                  ParameterOffers }

ParameterOffers ::= SEQUENCE
      { dimensionOffer    [0]  IMPLICIT BIT STRING
                                {  oneDimension          (0),
                                   twoDimensions         (1),
                                   threeDimensions       (2) } OPTIONAL,
                                -- a "1" value for a bit indicates an offer
        xParam            [1]  IMPLICIT  DimensionParamOffer  OPTIONAL,
        yParam            [2]  IMPLICIT  DimensionParamOffer  OPTIONAL,
        zParam            [3]  IMPLICIT  DimensionParamOffer  OPTIONAL,
        erasure           [4]  IMPLICIT  BIT  STRING  { yes (0), no (1) }  OPTIONAL,
        repertoire        [5]  IMPLICIT  CompoundRepertoireOffer  OPTIONAL,
        emphasis          [6]  IMPLICIT  CompoundEmphasisOffer  OPTIONAL,
        foreground        [7]  IMPLICIT  CompoundColourOffer  OPTIONAL,
        background        [8]  IMPLICIT  CompoundColourOffer  OPTIONAL,
        access            [9]  IMPLICIT  BIT  STRING
                                {  wavar                 (0),     -- a "1" value for a bit indicates an offer
                                   waci                  (1),
                                   waca                  (2) }  OPTIONAL,
                                        -- only waci and waca may occur together.
        blockParams       [10] IMPLICIT  BlockParamOffer  OPTIONAL,
        fieldParams       [11] IMPLICIT  FieldParamOffer  OPTIONAL,
        rippleCapability  [12] IMPLICIT  BIT STRING { yes [0], no [1] }  OPTIONAL }
                                                        -- a "1" value for a bit indicates an offer

DimensionParamOffer ::= SEQUENCE
      { bound             [0]  IMPLICIT SEQUENCE
                                {  unbounded             NULL  OPTIONAL,
                                   limit                 G.IntegerOffer  OPTIONAL }  OPTIONAL,
        addressing        [1]  IMPLICIT BIT STRING
                                {  noConstraint          (0),     -- a "1" value for a bit indicates an offer
                                   higherOnly            (1),
                                   notPermitted          (2) }  OPTIONAL,
        absolute          [2]   IMPLICIT BIT STRING  {  yes (0), no (1)  }  OPTIONAL,
                                                        -- a "1" value for a bit indicates an offer
```

```
        window          [3]  IMPLICIT SEQUENCE
                          {  unbounded          NULL OPTIONAL,
                             limit              G.IntegerOffer OPTIONAL } OPTIONAL }

CompoundRepertoireOffer ::= SEQUENCE
        { repertoireCapability   [0] IMPLICIT G.IntegerOffer OPTIONAL,
                                 [1] IMPLICIT SEQUENCE OF RepertoireFontOffer OPTIONAL }
                                       -- offers for positions in repertoire list

RepertoireFontOffer ::= CHOICE                    -- offer for single position in repertoire list:
        { NULL,                                   --    either place holder
          SEQUENCE OF SEQUENCE                    --    or list of alternatives
                  {   repertoire      [0]  IMPLICIT RepertoireAssignment OPTIONAL,
                      fontCapability  [1]  IMPLICIT G.IntegerOffer OPTIONAL,
                                      [2]  IMPLICIT SEQUENCE OF FontAssignment OPTIONAL } }

RepertoireAssignment ::= SEQUENCE
        { type                                    [0]  IMPLICIT OBJECT IDENTIFIER OPTIONAL,
          value     CHOICE {  iso2022             [1]  IMPLICIT SEQUENCE OF OCTET STRING,
                                                     -- the ESC character is not sent
                              isonnnn             [2]  ANY } }
        -- If "type" has the value {vt-b-rep-iso2022} or is not present, then the syntax for "value" is a sequence of
        --    up to 4 escape sequences as identified in ISO/IEC International Register of Coded Character Sets to be used
        --    with Escape Sequences identifying character repertoires.
        -- Otherwise, the syntax is ANY which may be resolved by reference to the definition specified in "type".

FontAssignment ::= SEQUENCE
        { type                                    [0]  IMPLICIT OBJECT IDENTIFIER OPTIONAL,
          value     CHOICE {  vtadhoc             [1]  IMPLICIT PrintableString,
                              isonnnn             [2]  ANY } }
        -- The "type" value for "FontAssignment" has a modal effect. It determines the CHOICE taken until another
        --  "type" value is encountered.
        -- If the modal "type" value is {vt-b-font-adhoc} or has not been set,
        --  then the syntax of "value" is a PrintableString.
        -- Otherwise, the syntax is ANY which may be resolved by reference to the definition specified in "type".

CompoundEmphasisOffer ::= SEQUENCE OF CHOICE
                          {  NULL,                              -- place holder
                             SEQUENCE OF PrintableString }      -- multiple offer of single emphasis subattribute

CompoundColourOffer ::= SEQUENCE
        { colourCapability  [0] IMPLICIT G.IntegerOffer OPTIONAL,
          colourValues      [1] IMPLICIT SEQUENCE OF ColourAssignment OPTIONAL }

ColourAssignment ::= SEQUENCE                       -- offer for a single position in a list
        { type              [0] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
          value CHOICE {    iso6429                 [1]  IMPLICIT  PrintableString,
                            isonnnn                 [2]  ANY } }
        -- The "type" value for "ColourAssignment" has a modal effect.
        -- It determines the CHOICE taken until another "type" value is encountered.
        -- If the modal "type" value is {vt-b-colour-iso6429} or has not been set,
        --    then the syntax of "value" is an English colour name to be found in ISO/IEC 6429.
        -- Otherwise, the syntax is ANY which may be resolved by reference to the definition specified in "type".
```

```
BlockParamOffer ::= SEQUENCE
    { capability        [0] IMPLICIT BIT STRING { yes (0), no (1) } OPTIONAL,
                                                    -- a "1" value for a bit indicates an offer
      bound             [1] IMPLICIT SEQUENCE
                          {     unbounded       NULL OPTIONAL,
                                limit           G.IntegerOffer OPTIONAL  } OPTIONAL  }

FieldParamOffer ::= SEQUENCE
    { capability        [0] IMPLICIT BIT STRING  { yes (0), no (1) } OPTIONAL,
                                                    -- a "1" value for a bit indicates an offer
      maxFields         [1] IMPLICIT SEQUENCE
                          {     unbounded       NULL OPTIONAL,
                                limit           G.IntegerOffer OPTIONAL  } OPTIONAL,
      maxFieldElements [2] IMPLICIT SEQUENCE
                          {     unbounded       NULL OPTIONAL,
                                limit           G.IntegerOffer OPTIONAL  } OPTIONAL,
      accessOutside     [3] IMPLICIT BIT STRING   { allowed (0), notAllowed (1) } OPTIONAL  }
                                                    -- a "1" value for a bit indicates an offer
```

12.3.3 CDS Values

```
                --*****************************************************  *
                --*          The following are for returned values during negotiation      *
                --*****************************************************  *
Values ::= SET OF SEQUENCE      {   name        PrintableString,
                                    ParameterValues  }

ParameterValues ::= SEQUENCE
    { dimension         [0]  IMPLICIT  INTEGER OPTIONAL,
      xParam            [1]  IMPLICIT  DimensionParamValues OPTIONAL,
      yParam            [2]  IMPLICIT  DimensionParamValues OPTIONAL,
      zParam            [3]  IMPLICIT  DimensionParamValues OPTIONAL,
      erasure           [4]  IMPLICIT  BOOLEAN OPTIONAL,     -- true="yes", false or absent ="no"
      repertoire        [5]  IMPLICIT  CompoundRepertoireValue OPTIONAL,
      emphasis          [6]  IMPLICIT  CompoundEmphasisValue OPTIONAL,
      foreground        [7]  IMPLICIT  CompoundColourValue OPTIONAL,
      background        [8]  IMPLICIT  CompoundColourValue OPTIONAL,
      access            [9]  IMPLICIT  INTEGER   { wavar (0), waci (1), waca (2 )  } OPTIONAL,
      blockParams      [10]  IMPLICIT  BlockParamValues OPTIONAL,
      fieldParams      [11]  IMPLICIT  FieldParamValues OPTIONAL,
      rippleCapability [12]  IMPLICIT BOOLEAN OPTIONAL      -- true = "yes", false or absent = "no"
                                                          }

DimensionParamValues ::= SEQUENCE
    { bound             [0]  CHOICE         {       unbounded     NULL,
                                                    limit         INTEGER  } OPTIONAL,
      addressing        [1]  IMPLICIT INTEGER
                          {  noConstraint           (0),
                             higherOnly             (1),
                             notPermitted           (2)  } OPTIONAL,
      absolute          [2]  IMPLICIT BOOLEAN OPTIONAL,              -- true="yes", false or absent ="no"
      window            [3]  CHOICE         {       unbounded     NULL,
                                                    limit          INTEGER  } OPTIONAL  }
```

```
CompoundRepertoireValue ::= SEQUENCE
        { repertoireCapability     [0]  IMPLICIT INTEGER OPTIONAL,
                                   [1]  IMPLICIT SEQUENCE OF RepertoireFontValue OPTIONAL  }

RepertoireFontValue ::= CHOICE
        { NULL,
          SEQUENCE
                {                          [0]   IMPLICIT RepertoireAssignment OPTIONAL,
                  fontCapability           [1]   IMPLICIT INTEGER OPTIONAL,
                                           [2]   IMPLICIT SEQUENCE OF FontAssignment OPTIONAL  }  }

CompoundEmphasisValue ::= SEQUENCE OF CHOICE
                        {     NULL,                  -- place holder
                              PrintableString        -- single acceptance value of emphasis subattribute
                              }

CompoundColourValue ::= SEQUENCE
        { capability           [0]  IMPLICIT INTEGER OPTIONAL,
          assignments          [1]  IMPLICIT SEQUENCE OF ColourAssignment OPTIONAL  }

BlockParamValues ::= SEQUENCE
        { capability           [0]  IMPLICIT BOOLEAN OPTIONAL,    -- true="yes", false or absent ="no"
          bound                [1]  CHOICE    {       unbounded    NULL,
                                                      limit        INTEGER  } OPTIONAL  }

FieldParamValues ::= SEQUENCE
        { capability           [0]  IMPLICIT BOOLEAN OPTIONAL,    -- true="yes", false or absent ="no"
          maxFields            [1]  CHOICE    {       unbounded    NULL,
                                                      limit        INTEGER  } OPTIONAL,
          maxFieldElements     [2]  CHOICE    {       unbounded    NULL,
                                                      limit        INTEGER  } OPTIONAL,
          accessOutside        [3]  IMPLICIT INTEGER
                               { allowed (0), notAllowed (1) } OPTIONAL  }     -- absence implies "allowed"

END -- of CDS Definitions
```

12.4 Control, Signal and Status definitions

```
CSS DEFINITIONS ::= BEGIN
```

12.4.1 CSS Identifiers

```
                --* ********************************************** *
                --*          The following are to identify parameters to be negotiated     *
                --* ********************************************** *

Identifier ::= SET OF SEQUENCE      {   name       PrintableString,
                                        ParameterIdents  }
```

```
ParameterIdents ::= SEQUENCE
        { typeIdentifier        [0] IMPLICIT NULL OPTIONAL,
          access                [1] IMPLICIT NULL OPTIONAL,
          trigger               [2] IMPLICIT NULL OPTIONAL,
          size                  [3] IMPLICIT NULL OPTIONAL,
          category              [4] IMPLICIT NULL OPTIONAL,
          repertoire            [5] IMPLICIT NULL OPTIONAL,
          priority              [6] IMPLICIT NULL OPTIONAL,
          structure             [7] IMPLICIT NULL OPTIONAL,
          multiElement          [8] IMPLICIT SEQUENCE OF ElementParamIdent OPTIONAL }

ElementParamIdent ::= SEQUENCE
        { elementIdentifier     [0] IMPLICIT INTEGER,
          size                  [3] IMPLICIT NULL OPTIONAL,
          category              [4] IMPLICIT NULL OPTIONAL,
          repertoire            [5] IMPLICIT NULL OPTIONAL }
```

12.4.2 CSS Offers

```
        --*****************************************************
        --*          The following are for offered values during negotiation        *
        --*****************************************************

Offer ::= SET OF SEQUENCE       {   name            PrintableString,
                                    ParameterOffers }

ParameterOffers ::= SEQUENCE
        { typeIdentifier        [0] IMPLICIT SET OF CHOICE  { OBJECT IDENTIFIER,  PrintableString },
          access                [1] IMPLICIT AccessRuleOffer OPTIONAL,
          trigger               [2] IMPLICIT BIT STRING     {      yes (0),  no (1) }  OPTIONAL,
          size                  [3] IMPLICIT G.IntegerOffer OPTIONAL,
          category              [4] IMPLICIT BIT STRING     {      character    (0),   -- value 1 for a bit implies
                                                                   boolean      (1),   --  offer, value 0 implies
                                                                   symbolic     (2),   --  no offer.
                                                                   integer      (3),
                                                                   transparent (4)  } OPTIONAL,
          repertoire            [5] IMPLICIT SEQUENCE OF  CDS.RepertoireAssignment OPTIONAL,
          priority              [6] IMPLICIT BIT STRING     {      normal  (0),    -- value 1 for a bit implies
                                                                   high      (1),   --   offer, value 0 implies no offer.
                                                                   urgent    (2)  } OPTIONAL,
          structure             [7] IMPLICIT SEQUENCE
                                    { nonParametric      NULL OPTIONAL,
                                      numberElements   G.IntegerOffer OPTIONAL  } OPTIONAL,
          multiElement          [8] IMPLICIT SEQUENCE OF ElementParamOffer OPTIONAL }
```

```
ElementParamOffer ::= SEQUENCE
        { elementIdentifier   [0] IMPLICIT INTEGER,
          size                [3] IMPLICIT G.IntegerOffer OPTIONAL,
          category            [4] IMPLICIT BIT STRING     {     character     (0),     -- value 1 for a bit implies
                                                                boolean       (1),     -- offer, value 0 implies
                                                                symbolic      (2),     -- no offer.
                                                                integer       (3),
                                                                transparent   (4) } OPTIONAL,
          repertoire          [5] IMPLICIT SEQUENCE OF CDS.RepertoireAssignment  OPTIONAL }

AccessRuleOffer ::= BIT STRING      {   wavar            (0),
                                        waci             (1),     -- value 1 for a bit implies
                                        waca             (2),     -- offer, value 0 implies
                                        nsac             (3),     -- no offer.
                                        wavar-and-waci   (4),
                                        wavar-and-waca   (5),
                                        no-access        (6) }
```

12.4.3 CSS Values

```
                --* ********************************************** *
                --*          The following are for returned values during negotiation      *
                --* ********************************************** *
Values ::= SET OF SEQUENCE      {   name          PrintableString,
                                    ParameterValues }
ParameterValues ::= SEQUENCE
        { typeIdentifier      [0] CHOICE      {       OBJECT IDENTIFIER,
                                                      PrintableString },
          access              [1] IMPLICIT  AccessRuleValue  OPTIONAL,
          triggerSelected     [2] IMPLICIT  BOOLEAN OPTIONAL,
          size                [3] IMPLICIT  INTEGER OPTIONAL,
          category            [4] IMPLICIT  INTEGER { character     (0),
                                                      boolean       (1),
                                                      symbolic      (2),
                                                      integer       (3),
                                                      transparent   (4) }   OPTIONAL,
          repertoire          [5] IMPLICIT  CDS.RepertoireAssignment OPTIONAL,
          priority            [6] IMPLICIT  INTEGER { normal        (0),
                                                      high          (1),
                                                      urgent        (2)   } OPTIONAL,
          structure           [7] CHOICE      {       nonParametric         NULL,
                                                      numberElements        INTEGER } OPTIONAL,
          multiElement        [8] IMPLICIT SEQUENCE OF ElementParamValue OPTIONAL }
                -- These values are responses to an offer and must be consistent with the proposals.

ElementParamValue ::= SEQUENCE
        { elementIdentifier   [0] IMPLICIT INTEGER,
          size                [3] IMPLICIT INTEGER OPTIONAL,
          category            [4] IMPLICIT INTEGER { character     (0),
                                                     boolean       (1),
                                                     symbolic      (2),
                                                     integer       (3),
                                                     transparent   (4) } OPTIONAL,
          repertoire          [5] IMPLICIT CDS.RepertoireAssignment  OPTIONAL }
```

```
AccessRuleValue ::= INTEGER     {   wavar            (0),
                                    waci             (1),
                                    waca             (2),
                                    nsac             (3),
                                    wavar-and-waci   (4),
                                    wavar-and-waca   (5),
                                    no-access        (6)  }

END -- of CSS Definitions
```

12.5 Device Object definitions

DEV DEFINITIONS ::= BEGIN

12.5.1 DEV Identifiers

```
--*****************************************************  *
--*          The following are to identify parameters to be negotiated     *
--*****************************************************  *

Identifier ::= SET OF SEQUENCE { name           PrintableString,
                                 ParameterIdents }

ParameterIdents ::= SEQUENCE
    { defaultCOaccess           [0]  IMPLICIT NULL OPTIONAL,
      defaultCOPriority         [1]  IMPLICIT NULL OPTIONAL,
      deviceRepertoire          [2]  IMPLICIT CDS.CompoundRepertoireIdent  OPTIONAL,
      deviceEmphasis            [3]  IMPLICIT CDS.CompoundEmphasisIdent  OPTIONAL,
      deviceForeground          [4]  IMPLICIT CDS.CompoundColourIdent OPTIONAL,
      deviceBackground          [5]  IMPLICIT CDS.CompoundColourIdent OPTIONAL,
          -- Tags 2 to 5 above make cross references to the definitions in the CDS module.
          -- Not all the components of those definitions can be used validly in this context.
          -- In particular, the components relating to the lengths of lists (capabilities) shall not be present
      minimumXarrayLength       [6]  IMPLICIT NULL OPTIONAL,
      minimumYarrayLength       [7]  IMPLICIT NULL OPTIONAL,
      deviceControlObjectNames  [8]  IMPLICIT NULL OPTIONAL,
      deviceDisplayObjectName   [9]  IMPLICIT NULL OPTIONAL,
      terminationEventList      [10] IMPLICIT NULL OPTIONAL,
      terminationLength         [11] IMPLICIT NULL OPTIONAL,
      terminationTimeout        [12] IMPLICIT NULL OPTIONAL,
      defaultCOtrigger          [13] IMPLICIT NULL OPTIONAL,
      defaultCOinitialValue     [14] IMPLICIT NULL OPTIONAL  }
```

12.5.2 DEV Offers

```
        --* ********************************************** *
        --*          The following are for offered values during negotiation      *
        --* ********************************************** *
Offer ::= SET OF SEQUENCE     {  name              PrintableString,
                                 ParameterOffers }

ParameterOffers ::= SEQUENCE
    { defaultCOaccess              [0]  IMPLICIT  CSS.AccessRuleOffer  OPTIONAL,
      defaultCOpriority            [1]  IMPLICIT BIT STRING {  normal  (0), -- bit value 1 implies offer,
                                                               high    (1), --  value 0 implies no offer.
                                                               urgent  (2) } OPTIONAL,
      deviceRepertoireAssignment   [2]  IMPLICIT  CDS.CompoundRepertoireOffer  OPTIONAL,
      deviceEmphasisAssignment     [3]  IMPLICIT  CDS.CompoundEmphasisOffer  OPTIONAL,
      deviceForegroundAssignment   [4]  IMPLICIT  CDS.CompoundColourOffer  OPTIONAL,
      deviceBackgroundAssignment   [5]  IMPLICIT  CDS.CompoundColourOffer  OPTIONAL,
                                          -- see note in 12.5.1
      minimumXarrayLength          [6]  IMPLICIT  G.IntegerOffer  OPTIONAL,
      minimumYarrayLength          [7]  IMPLICIT  G.IntegerOffer  OPTIONAL,
      deviceControlObjectNames     [8]  IMPLICIT SEQUENCE OF
                                            SEQUENCE OF PrintableString OPTIONAL,
      deviceDisplayObjectName      [9]  IMPLICIT SEQUENCE OF  PrintableString OPTIONAL,
      terminationEventList         [10] IMPLICIT SEQUENCE OF  SET OF SEQUENCE
                        {   event                 ANY,
                            eventId               SEQUENCE { G.IntegerOffer OPTIONAL,
                                                             NULL   OPTIONAL } } OPTIONAL,
                        -- The datatypes for "event" will be as defined in the vte-profile used as
                        --    the initial draft-VTE for the individual instance of negotiation.
      terminationLength            [11] IMPLICIT SEQUENCE
                                        {   length          [0] IMPLICIT G.IntegerOffer,
                                            eventId         [1] IMPLICIT SEQUENCE
                                                                { G.IntegerOffer OPTIONAL,
                                                                  NULL  OPTIONAL } } OPTIONAL,
      terminationTimeout           [12] IMPLICIT SEQUENCE
                                            {               TimeOffer,
                                              eventId       SEQUENCE
                                                            { G.IntegerOffer   OPTIONAL,
                                                              NULL  OPTIONAL } } OPTIONAL,
      defaultCOtrigger             [13] IMPLICIT BIT STRING {   notSelected     (0),
                                                                selected        (1) } OPTIONAL,
      defaultCOinitialValue        [14] IMPLICIT SEQUENCE OF SEQUENCE
                                        { value     [0] IMPLICIT BIT STRING,
                                          mask      [1] IMPLICIT BIT STRING OPTIONAL }
                                           -- See note under mask in G.COUpdate
                                                            OPTIONAL }

TimeOffer ::= SET OF CHOICE
    { value             [0] IMPLICIT SEQUENCE  {      multiplier INTEGER,
                                                      exponent INTEGER },
      range             [1] IMPLICIT SEQUENCE  {      lowerMultiplier     INTEGER,
                                                      lowerExponent       INTEGER,
                                                      upperMultiplier     INTEGER,
                                                      upperExponent       INTEGER } }
```

12.5.3 DEV Values

```
--*****************************************************
--*          The following are for returned values during negotiation      *
--*****************************************************

Values ::= SET OF SEQUENCE      {   name          PrintableString,
                                    ParameterValues }

ParameterValues ::= SEQUENCE
      { defaultCOaccess                 [0]  IMPLICIT CSS.AccessRuleValue OPTIONAL,
        defaultCOpriority               [1]  IMPLICIT INTEGER    {   normal  (0),
                                                                     high    (1),
                                                                     urgent  (2) }  OPTIONAL,
        deviceRepertoireAssignment      [2]  IMPLICIT  CDS.CompoundRepertoireValue OPTIONAL,
        deviceEmphasisAssignment        [3]  IMPLICIT  CDS.CompoundEmphasisValue OPTIONAL,
        deviceForegroundAssignment      [4]  IMPLICIT  CDS.CompoundColourValue OPTIONAL,
        deviceBackgroundAssignment      [5]  IMPLICIT  CDS.CompoundColourValue OPTIONAL,
                                             --see note in 12.5.1
        minimumXarrayLength             [6]  IMPLICIT  INTEGER OPTIONAL,
        minimumYarrayLength             [7]  IMPLICIT  INTEGER OPTIONAL,
        deviceControlObjectNames        [8]  IMPLICIT  SEQUENCE  OF PrintableString OPTIONAL,
        deviceDisplayObjectName         [9]  IMPLICIT  PrintableString OPTIONAL,
        terminationEventList            [10] IMPLICIT  SET OF SEQUENCE
                                                  {     event ANY,
                                                        eventId CHOICE { INTEGER, NULL}  } OPTIONAL,
                        -- The datatypes for "event" will be as defined in the vte-profile used as the initial
                        --  draft-VTE for the individual instance of negotiation.
        terminationLength               [11] IMPLICIT SEQUENCE
                                                  {     length          [0] IMPLICIT INTEGER OPTIONAL,
                                                        eventID CHOICE { [1] IMPLICIT INTEGER,
                                                                         [2] IMPLICIT NULL } } OPTIONAL,
        terminationTimeout              [12] IMPLICIT SEQUENCE
                                                  {     timeMultiplier      INTEGER,
                                                        timeExponent        INTEGER,
                                                        eventId   CHOICE { INTEGER, NULL } }  OPTIONAL,
        defaultCOtriggerSelected        [13] IMPLICIT BOOLEAN OPTIONAL,

        defaultCOinitialValue           [14] IMPLICIT SEQUENCE
                              { value                   [0] IMPLICIT BIT STRING,
                                mask                    [1] IMPLICIT BIT STRING OPTIONAL } OPTIONAL  }
                                        -- See note under mask above.

END -- of DEV Definitions
```

13 Conformance

13.1 Dynamic conformance requirements

A system claiming conformance to this part of ISO/IEC 9041 shall exhibit external behaviour consistent with having implemented all of the following:

a) a virtual terminal protocol machine as defined by annex A of this part of ISO/IEC 9041;

b) the use of the Association Control Service Element and of the Presentation Layer as defined in clause 11 of this part of ISO/IEC 9041;

c) encoding of virtual terminal protocol data units as defined in clause 12 of this part of ISO/IEC 9041.

13.2 Static conformance requirements

A system claiming conformance to this part of ISO/IEC 9041 shall

a) support the A-mode or the S-mode or both;

b) support the VT environment defined by the A-mode default profile if A-mode is supported;

c) support the VT environment defined by the S-mode default profile if S-mode is supported;

d) be capable of requesting a VT-association or of responding to a VT-association attempt or both;

e) be capable of accepting all correct sequences of VT protocol elements received from peer equipment and of responding with correct VT protocol element sequences for the defined states of the VT-association;

f) be capable of responding to invalid events as specified in A.4.1;

g) support

1) character repertoire assignment as specified in 18.2.4 and table 4 of ISO/IEC 9040,

2) any character set designated for inclusion in a repertoire where negotiation of the repertoire between the VT-users is successful,

3) transmission and reception of any sequence of characters from each character set supported, encoded in accordance with its registered details as a value of octet-string type and then used as an element of the abstract syntax defined in clause 12.

A system claiming conformance to this part of ISO/IEC 9041 shall be capable of the following dependency requirements:

h) supporting the Association Control Service Element in the Application Layer;

i) supporting the Presentation kernel and Session kernel functional units;

j) supporting other Session functional units as defined in 11.1.1 d) of this part of ISO/IEC 9041;

k) supporting the ASN.1 Basic Encoding Rules defined in ISO/IEC 8825 even if the system supports alternative encodings.

13.3 Protocol Implementation Conformance Statement (PICS)

The supplier of a protocol implementation which is claimed to conform to this International Standard shall complete a copy of the PICS proforma provided in Part 2 Annex A of this International Standard, and shall provide the information necessary to identify both the supplier and the implementation.

Annex A
(normative)
State Tables

A.1 General

This annex describes the Basic Class Virtual Terminal Protocol in terms of State Tables. The state tables show the state of a Virtual Terminal association, the events that occur in the protocol and indicate the actions to be taken by the VTPM in response to these events. These state tables do not constitute a full formal definition of the Basic Class Virtual Terminal Protocol; they are included to provide a more precise specification of the procedures defined in clauses 7 through 10. In the case of arbitration or dispute, this annex takes precedence over clauses 7 through 10.

The state tables are supported by:

a) the definition of VTE -parameter status and variables in A.2;

b) table A.1 which specifies the acronym, category and description of each incoming event. The categories are VT-user event, VTP-element event, other service provider event and VTPM event;

c) table A.2 which specifies the identifier and description of each state;

NOTE – Suffixes to the identifiers in table A.2 have the following meanings:

B indicates A-mode
N indicates S-mode, WAVAR not held
Q indicates A-mode
R indicates A-mode
S indicates S-mode
T indicates S-mode, WAVAR held

d) table A.3 which specifies the acronym, category and description of each outgoing event. The categories are VT-user event and VTP-element event;

e) table A.4 which specifies the atomic actions referenced by the state tables;

f) table A.5 which specifies the predicates referenced in subsequent action tables.

The state tables themselves are presented in tables A.6 through A.15. These tables are to be interpreted in pairs. The first table of each pair relates the states to incoming events and references an entry in the second table. The first table is formally referred to as the state table and the second table as the corresponding action list. The number of states make it inappropriate to describe the protocol in a single table. Therefore, the states are separated into sectors as follows:

g) Sector 1, Association Control & Quiescent;

h) Sector 2, Negotiation Initiation;

i) Sector 3, Negotiation;

j) Sector 4, Data Transfer;

k) Sector 5, Exceptions.

Actions other than the common actions are listed and numbered separately for each sector.

A.2 Parameters, VTPM rights and variables

A.2.1 Parameters

A VTE comprises a set of instances of the VTE-parameters defined in ISO/IEC 9040; each member of the set has three associated status variables which are orthogonal in nature and may have a value.

The Existence status has two values, Defined and Undefined.

The Negotiation status has four values:

a) Idle;
b) Invited;
c) Offered;
d) Counter Offered.

The Last Action indicator has two values, Local and Peer.

If a parameter instance has an Existence Status of Defined then a single value as defined in ISO/IEC 9040 for the parameter type is assigned to it.

The draft-VTE (which exists only during association establishment and during negotiation) contains parameter instances which may have any of the above categories of status, subject to the functional units which have been selected.

Figure A.1 shows the state transitions for the Negotiation Status of any one VTE-parameter during Multiple Interaction Negotiation.

The Last Action indicator is used in each Negotiation State except Idle to enable proper sequence to be policed (see 30.3 of ISO/IEC 9040).

The current-VTE (which exists at all times after a VT-association has been set up) is restricted to parameter instances which are either Undefined or Defined.

A full-VTE is one in which all parameter instances have Defined status.

In order for a draft-VTE to become a current-VTE after an instance of MIN, the Negotiating status of each parameter must be Idle.

For data transfer to take place the current-VTE must be a full-VTE, and must also be agreed upon. It is this condition which is recorded by the variable Vena (see A.2.2).

A.2.2 Variables

The following variables are maintained by each VTPM. These may be referenced in predicates, see clause A.3 and table A.6. Some variables are set explicitly by actions in the state tables; others are set implicitly as detailed below. For each variable, the type is specified as integer, boolean, or enumerated. For variables of enumerated type, the possible values are specified.

Each variable may also have the attributes 'parameter' or 'permanent'. A variable with the attribute 'parameter' is implicitly set whenever an incoming event (VTP element or from VT-user) is received by the VTPM. Any outgoing event has its parameters set from the corresponding VTPM parameter variables.

A variable which does not have the attribute 'permanent' is only significant during the processing of the event which set the variable; variables with the attribute 'permanent' must be saved by the VTPM between events.

A.2.2.1 Vcho : enumerated, parameter, permanent

This is associated with VT service parameter VT-vte-choice. It may take one of the values dr = "draft", cu = "current", or ei = "either".

A.2.2.2 Vena : boolean, permanent

True if a current-VTE has ever been agreed. This variable is always set explicitly by state table actions (including SetCu-VTE).

NOTE – Vena implies that the current-VTE is a full-VTE, the reverse implication does NOT hold.

A.2.2.3 Vns : integer, permanent

The number of NDQs held by a VTPM awaiting transmission to the peer VTPM. This variable is always set explicitly by state table actions.

A.2.2.4 Vnt : integer, permanent

The number of VT service indications held by a VTPM awaiting delivery to the local VT-service user. This variable is always set explicitly by state table actions.

A.2.2.5 Vqdl : boolean, permanent

True if a DLQ with acknowledgement has been received but not passed to the VT-user. It is always explicitly set by state table actions.

A.2.2.6 Vra : boolean, parameter

This is associated with the VT service parameter VT-ack-request. It is true if and only if VT-ack-request = "acknowledgement".

A.2.2.7 Vrcl : boolean, permanent

True if there has been a collision of RLQs. It is always explicitly set by state table actions.

NOTE – this can only occur when VT-token is held by both sides of the dialogue.

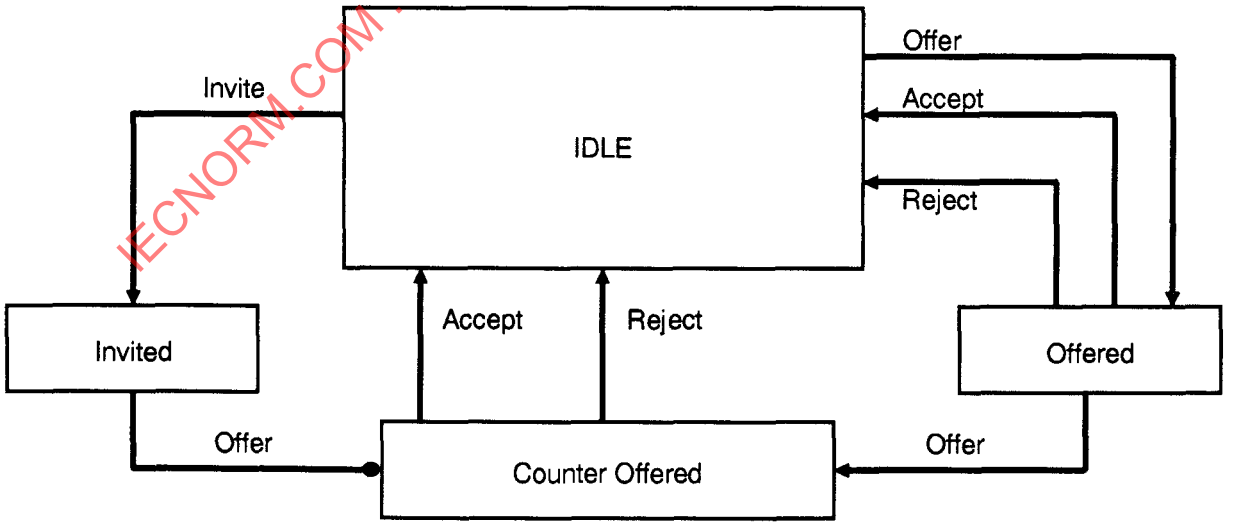

Figure A.1 - Negotiation state diagram

A.2.2.8 Vrea : enumerated, parameter

This is associated with the VT service parameter VT-provider-failure-reason. It may take one of the values col = "collision detected", pns = "profile not supported" or "parameter not supported" or "parameter combination not supported", ein = "VTE-incomplete", or other values.

A.2.2.9 Vrsl : enumerated, parameter

This is associated with the service parameter VT-result. It may take one of the values scc = "success", fai = "failure" and sww ="success-with-warning".

A.2.2.10 Vsmd : boolean, permanent

True in S-mode and false in A-mode. It is set implicitly from the selected profile when ASR, SPR, SNR, ENR or BKR is sent or received.

A.2.2.11 Vtkp : enumerated, parameter

In S-mode, this is associated with the VT service parameter VT-token. In A-mode, it is always set explicitly by state table actions. It may take one of the values ini = "initiator", acc = "acceptor", or cho = "acceptor choice".

A.2.2.12 Vtok : boolean, permanent

Vtok is a boolean which records whether the VT-token is held. If VT-token does not exist this boolean is always true. It is a protocol error if not all of the available Session tokens are on the same side of the dialogue. Vtok is always set implicitly.

A.2.2.13 Vbrkp : enumerated, parameter

This is associated with the VT service parameter VT-information issued with a VT-BREAK request.

A.2.2.14 Vbrks : enumerated, permanent

This is used to store the VT service parameter VT-information of a colliding VT-BREAK request during the exception report procedure.

A.3 Conventions for use of state tables

A.3.1 Incoming events are represented in the state tables by their acronyms as specified in table A.1.

A.3.2 The states are represented in the state tables by identifiers as specified in table A.2.

A.3.3 Each intersection of a state and event which is always invalid is left blank. A (state,event) pair is either valid or invalid depending upon the predicates in its associated action.

A.3.4 At each intersection that is valid, the state tables specify an action number which refers to the list of actions in the table immediately following it or to the common actions in table A.7.

A.3.5 The entries in the list of actions consist of either

a) an unconditional action list containing a list of zero or more entries taken from tables A.3, A.4, and A.7 separated by ";" and optionally followed by the symbol "⇒" followed by the identifier of the resultant state; if no state change is to occur the symbol "SAMESTATE" replaces the identifier; an x or a y in the identifier indicates the same value as that appertaining to the input state number; or

b) one or more conditional action lists each containing
   1) a predicate expression, consisting of entries from table A.5 and the boolean operators AND "&", OR "+" and NOT "~", terminated by a colon character,
   2) an unconditional action list as in a).

Where a conditional action list occupies more than one line, subsequent lines are inset.

A.4 Actions to be taken by the VTPM

The state tables refer to the actions to be taken by the VTPM.

A.4.1 Invalid intersections

If the incoming event is invalid for the current state, one of the following shall be done:

a) for an event relating to the VT service (i.e., one coming from the VT-user) take local action which is outside the scope of this part of ISO/IEC 9041, to recover from the error;

b) for an event relating to a received protocol element, follow the procedure for the treatment of protocol errors (see 5.5.7) if the state of the supporting presentation connection makes this possible;

c) for events falling into neither of the above categories take no action.

A.4.2 Valid intersections

For each predicate expression in turn which is true, the VTPM shall take the specific actions in the order given until the list is exhausted or a state change is indicated (including ⇒ SAMESTATE). The VTPM shall change to the resultant state identified after the symbol "⇒". The absence of a predicate expression is equivalent to the presence of a predicate expression with the value "true". If none of the predicate expressions are true, the VTPM shall consider the intersection to be invalid (see A.4.1). An incoming event corresponding to a functional unit not successfully negotiated shall be considered by the VTPM to be invalid.

Table A.1 - Incoming Events

Acronym	Category	Description
VASSreq	VT-user	VT-ASSOCIATE request
VASSrsp	VT-user	VT-ASSOCIATE response
VBRKreq	VT-user	VT-BREAK request
VBRKrsp	VT-user	VT-BREAK response
VDACKreq	VT-user	VT-ACK-RECEIPT request
VDATreq-u	VT-user	VT-DATA request addressing urgent priority CO
VDATreq-h	VT-user	VT-DATA request addressing high priority CO
VDATreq-n	VT-user	VT-DATA request addressing normal priority CO
VDATreq-sqtr	VT-user	VT-DATA request addressing trigger control CO
VDELreq	VT-user	VT-DELIVER request
VENEGreq	VT-user	VT-END-NEG request
VENEGrsp	VT-user	VT-END-NEG response
VGVTreq	VT-user	VT-GIVE-TOKENS request
VNACCreq	VT-user	VT-NEG-ACCEPT request
VNINVreq	VT-user	VT-NEG-INVITE request
VNOFFreq	VT-user	VT-NEG-OFFER request
VNREJreq	VT-user	VT-NEG-REJECT request
VRELreq	VT-user	VT-RELEASE request
VRELrsp	VT-user	VT-RELEASE response
VRQTreq	VT-user	VT-REQUEST-TOKENS request
VSNEGreq	VT-user	VT-START-NEG request
VSNEGrsp	VT-user	VT-START-NEG response
VSWPreq	VT-user	VT-SWITCH-PROFILE request
VSWPrsp	VT-user	VT-SWITCH-PROFILE response
VUABreq	VT-user	VT-U-ABORT request
APQ	VTP element	VT-P-ABORT-REQ
ASQ	VTP element	VT-ASSOCIATE-REQ
ASR	VTP element	VT-ASSOCIATE-RESP
AUQ	VTP element	VT-U-ABORT-REQ
BKQ	VTP element	VT-BREAK-REQ
BKR	VTP element	VT-BREAK-RESP
DAQ	VTP element	VT-ACK-RECEIPT
DLQ	VTP element	VT-DELIVER
ENQ	VTP element	VT-END-NEG-REQ
ENR	VTP element	VT-END-NEG-RESP
EXQ	VTP element	VT-P-EXCEPTION-REQ
EXR	VTP element	VT-P-EXCEPTION-RESP
GTQ	VTP element	VT-GIVE-TOKEN
HDQ	VTP element	VT-HIGH-PRI-DATA
NAQ	VTP element	VT-NEG-ACCEPT-REQ
NDQ-ntr	VTP element	VT-DATA not including update to trigger control object
NDQ-tr	VTP element	VT-DATA including update to trigger control object
NJQ	VTP element	VT-NEG-REJECT-REQ
NIQ	VTP element	VT-NEG-INVITE-REQ
NOQ	VTP element	VT-NEG-OFFER-REQ
RLQ	VTP element	VT-RELEASE-REQ
RLR	VTP element	VT-RELEASE-RESP
RTQ	VTP element	VT-REQUEST-TOKEN

Table A.1 - Incoming Events (concluded)

Acronym	Category	Description
SPQ	VTP element	VT-SWITCH-PROFILE-REQ
SPR	VTP element	VT-SWITCH-PROFILE-RESP
SNQ	VTP element	VT-START-NEG-REQ
SNR	VTP element	VT-START-NEG-RESP
UDQ	VTP element	VT-URGENT-DATA
autonomous	VTPM	Internal event not defined by VTP specification
exception	VTPM	internal exception event
VTAB	VTPM	Irrecoverable exception condition
PAB	Supporting service	Failure indication

Table A.2 - States

Identifier	Description
01	No association
02B,02S	Associate - awaiting target
03B,03S	Associate - awaiting user
10B,10N,10T	Environment not agreed
20B	Switch-profile, awaiting tokens
21Q	Switch-profile, awaiting peer
21R	Switch-profile, awaiting user
21N	Switch-profile, awaiting user
21T	Switch-profile, awaiting peer
30B	Multiple-interaction negotiation, awaiting tokens at start
31Q	Multiple-interaction negotiation, awaiting peer at start
31R	Multiple-interaction negotiation, awaiting user at start
31N	Multiple-interaction negotiation, awaiting user at start
31T	Multiple-interaction negotiation, awaiting peer at start
35Q	Multiple-interaction negotiation initiator, negotiating
36R	Multiple-interaction negotiation responder, negotiating
36N,36T	Multiple-interaction negotiation, negotiating
37N,37Q,37R,37T	Multiple-interaction negotiation, terminating
40N,40T	Data transfer
400B	Data transfer
42T	Data transfer, awaiting ack from peer
420B	Data transfer, awaiting ack from peer
42N	Data transfer, awaiting ack from user
402B	Data transfer, awaiting ack from user
422B	Data transfer, awaiting ack in both directions
50B	Release, awaiting tokens
51N	Release, awaiting user
51T	Release, awaiting peer
51Q	Release, awaiting peer
51R	Release, awaiting user
61	Break request received from user
62	Break request received from peer
71	Exception – await user
72	Exception – await user
73	Exception – await peer – storing break

Table A.3 - Outgoing events

Acronym	Category	Description
VACKind	VT-user	VT-ACK-RECEIPT indication
VASSind	VT-user	VT-ASSOCIATE indication
VASScnf	VT-user	VT-ASSOCIATE confirm
VBRKind	VT-user	VT-BREAK indication
VBRKcnf	VT-user	VT-BREAK confirm
VDATind-h	VT-user	VT-DATA indication, high object(s)
VDATind-n	VT-user	VT-DATA indication, normal object(s)
VDATind-n(Vnt)	VT-user	Sequence of Vnt VT-DATA indication, normal objects
VDATind-u	VT-user	VT-DATA indication, urgent object(s)
VDELind	VT-user	VT-DELIVER indication
VENEGind	VT-user	VT-END-NEG indication
VENEGcnf	VT-user	VT-END-NEG confirm
VGVTind	VT-user	VT-GIVE-TOKENS indication
VNINVind	VT-user	VT-NEG-INVITE indication
VNOFFind	VT-user	VT-NEG-OFFER indication
VNACCind	VT-user	VT-NEG-ACCEPT indication
VNREJind	VT-user	VT-NEG-REJECT indication
VPABind	VT-user	VT-P-ABORT indication
VRELind	VT-user	VT-RELEASE indication
VRELcnf	VT-user	VT-RELEASE confirm
VRQTind	VT-user	VT-REQUEST-TOKENS indication
VSNEGind	VT-user	VT-START-NEG indication
VSNEGcnf	VT-user	VT-START-NEG confirm
VSWPind	VT-user	VT-SWITCH-PROFILE indication
VSWPcnf	VT-user	VT-SWITCH-PROFILE confirm
VUABind	VT-user	VT-U-ABORT indication
VPEXind	VT-user	VT-P-EXCEPTION indication
APQ	VTP element	VT-P-ABORT-REQ
ASQ	VTP element	VT-ASSOCIATE-REQ
ASR	VTP element	VT-ASSOCIATE-RESP
AUQ	VTP element	VT-U-ABORT-REQ
BKQ	VTP element	VT-BREAK-REQ
BKR	VTP element	VT-BREAK-RESP
DAQ	VTP element	VT-ACK-RECEIPT
DLQ	VTP element	VT-DELIVER
ENQ	VTP element	VT-END-NEG-REQ
ENR	VTP element	VT-END-NEG-RESP
EXQ	VTP element	VT-P-EXCEPTION-REQ
EXR	VTP element	VT-P-EXCEPTION-RESP
GTQ	VTP element	VT-GIVE-TOKEN
HDQ	VTP element	VT-HIGH-PRI-DATA
NAQ	VTP element	VT-NEG-ACCEPT-REQ
NDQ-ntr	VTP element	VT-DATA not including update to trigger control object
NDQ-tr	VTP element	VT-DATA including update to trigger control object
NDQ(Vns)	VTP element	Sequence containing Vns NDQ-ntr, or Vns-1 NDQ-ntr followed by an NDQ-tr
NIQ	VTP element	VT-NEG-INVITE-REQ
NJQ	VTP element	VT-NEG-REJECT-REQ
NOQ	VTP element	VT-NEG-OFFER-REQ
RLQ	VTP element	VT-RELEASE-REQ
RLR	VTP element	VT-RELEASE-REQ
RTQ	VTP element	VT-REQUEST-TOKEN