

INTERNATIONAL STANDARD

ISO/IEC
9040

Second edition
1997-06-15

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

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



Reference number
ISO/IEC 9040:1997(E)
Documents



Document Sharing Hub

Contents

1 Scope..... 1

2 Normative references..... 1

3 Definitions..... 2

 3.1 Global OSI definitions..... 2

 3.2 Association Control Service Element definitions..... 2

 3.3 Virtual Terminal Service definitions..... 2

4 Abbreviations..... 5

 4.1 General..... 5

 4.2 Modes of operation..... 5

 4.3 VTE model components..... 5

 4.4 Access-rules..... 6

5 Conventions..... 6

6 General features..... 6

 6.1 Introduction..... 6

 6.2 Features of the Virtual Terminal Basic Class Service..... 6

 6.3 VT Environment (VTE) and VTE-parameters..... 7

 6.4 Virtual Terminal Environment Profiles..... 7

 6.5 Dialogue Control..... 7

7 Communication facilities..... 7

 7.1 Establishment facility..... 7

 7.2 Termination facility..... 7

 7.3 Negotiation facility..... 7

 7.4 Data Transfer facility..... 7

 7.5 Delivery Control facility..... 7

 7.6 Dialogue Management facility..... 7

 7.7 Interrupt facilities..... 7

 7.8 Exception Reporting facility..... 7

© ISO/IEC 1997

All rights reserved. Unless otherwise specified no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case Postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

8 Modes of operation	8
8.1 S-mode	8
8.2 A-mode	8
9 Access-rules	8
10 VT functional units	9
10.1 Negotiation functional units	9
10.2 Negotiated Release functional unit	10
10.3 Urgent Data functional unit	10
10.4 Break functional unit	10
10.5 Enhanced Access-rules functional unit	10
10.6 Structured Control Objects functional unit	10
10.7 Blocks functional unit	10
10.8 Fields functional unit	10
10.9 Reference Information Objects functional unit	10
10.10 Ripple functional unit	10
10.11 Exceptions functional unit	10
10.12 Context Retention functional unit	11
11 VT Environment Profiles (VTE-profiles)	11
12 The VTE Model	11
13 Display Objects	12
13.1 Structure	12
13.2 Attributes	16
14 Control objects	16
14.1 CO structure facilities and related restrictions	17
14.2 Standard control objects for fields and controlled data entry	17
14.3 Standard control object for dynamic termination conditions	18
14.4 Standard control object for notifying termination	18
14.5 Standard control object for echo control	18
14.6 Standard control object for ripple mode editing	18
15 Reference Information Objects	18
15.1 Structure	18
15.2 Generation and use of RIOs	18
16 Device objects	19
17 VTE Parameters and directed graph	19
17.1 Directed graph of VTE-parameters	19
17.2 VTE consistency rules	19
18 Display Object VTE-parameters	22
18.1 Primary VTE-parameters	22
18.2 Secondary VTE-parameters	22
18.3 Tertiary VTE-parameter	25
19 Operations on display objects	26
19.1 Addressing operations	26
19.2 Ripple operations	30
19.3 Logical ripple operations	31
19.4 Update operations	31
19.5 Access control over display object	38

20 Control object VTE-parameters.	39
20.1 Usage and effects of control object VTE-parameters	40
20.2 Usage and effects of data element VTE-parameters.	41
20.3 Standard control objects.	41
21 Reference Information Object VTE-parameters.	46
21.1 Availability.	46
21.2 VTE-parameters for RIOs.	46
22 Operations on RIOs	47
22.1 Availability.	47
22.2 Identification of RIO and RIO records	47
22.3 RIO update operations	47
22.4 RIO reference operations	47
23 Device object VTE-parameters	48
23.1 Default control object VTE-parameters	48
23.2 Minimum Length VTE-parameters	48
23.3 Device object VTE-parameters for Attributes.	48
23.4 Termination VTE-parameters.	49
23.5 Interaction between use of TCCO or FDCO and device object VTE-parameters.	50
24 Delivery control, synchronisation and net-effecting.	50
24.1 No delivery-control	50
24.2 Simple delivery-control	51
24.3 Quarantine delivery-control	51
24.4 Implicit delivery	51
24.5 Update queues and priority handling	51
25 Communication Model.	53
26 VT Services	53
27 VT service sequences	55
27.1 Phases	55
27.2 Phase transitions	55
27.3 Ownership of the WAVAR access-right	56
27.4 Availability and usage conditions of VT services	56
27.5 Service collisions in A-mode	57
28 Establishment facility.	57
28.1 VT-ASSOCIATE service.	57
29 Termination facility.	60
29.1 Services	60
29.2 VT-RELEASE service.	60
29.3 VT-U-ABORT service	60
29.4 VT-P-ABORT service	61
30 Negotiation facilities.	61
30.1 Switch Profile negotiation	61
30.2 Multiple interaction negotiation.	62
30.3 Sequence control for multiple interaction negotiation	65
31 Data Transfer facility	66
31.1 VT-DATA service	66

32 Delivery Control facility	67
32.1 VT-DELIVER service	67
32.2 VT-ACK-RECEIPT service	68
33 Dialogue Management facility	68
33.1 VT-GIVE-TOKENS service	68
33.2 VT-REQUEST-TOKENS service	68
34 Destructive Interrupt facility	68
34.1 VT-BREAK service	68
35 Exception reporting facility	69
35.1 VT-P-EXCEPTION service	69
Annex A Default VTE-profiles	71
A.1 Introduction to VTE-profile definitions	71
A.2 Notation for definition of VTE-profiles	71
A.3 S-mode Default VTE-profile, vt-b-spr-sd	72
A.4 A-mode Default VTE-profile, vt-b-spr-ad	72
Annex B Explanatory notes	74
B.1 Types of VT communication supported	74
B.2 Aid to understanding the role of display objects	74
B.3 Relation of update-window to buffering	74
B.4 Control object semantics	74
B.5 Echoing	74
B.6 Echo control	74
B.7 Echo control algorithm	75
B.8 Termination conditions	75
B.9 Synchronisation of update delivery	75
B.10 Multiple Interaction Negotiation	76
B.11 Semantics of display objects	76
B.12 Repertoires	76
B.13 Use of ISO 6429 Additional Controls in repertoires	76
B.14 Font Assignment VTE-parameters	77
B.15 Net-effecting	77
B.16 Interrupt facilities	77
B.17 Emphasis attribute	77
B.18 Supplementary explanatory material on field facilities	79
B.19 Supplementary explanatory material on block facilities	83
Annex C ASN.1 OBJECT IDENTIFIER values	85
C.1 For identification of this International Standard	85
C.2 For identification of attribute assignment types	85
Annex D Size of urgent control objects	86

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/IEC 9040 was prepared by Joint Technical Committee ISO/IEC JTC1, *Information technology*, Subcommittee SC21, *Open systems interconnection, data management and open distributed processing*.

This second edition cancels and replaces the first edition (ISO 9040:1990), which has been technically revised. It also incorporates Amendment 2:1992, Technical Corrigendum 1:1991, Technical Corrigendum 2:1992 and Technical Corrigendum 3:1993.

Annexes A and C form an integral part of this International Standard. Annexes B and D are for information only.

Introduction

This International Standard is one of a set of 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 into a series of layers of specification, each of manageable size.

The purpose of this International Standard (ISO/IEC 9040) is to define the service provided in the Application Layer by the Virtual Terminal (VT) Basic Class Service.

The Virtual Terminal Basic Class Service is provided by the Virtual Terminal Basic Class Protocol specified in ISO/IEC 9041 and making use of services available from the Association Control Service Element (ACSE) in the Application Layer and the Presentation Service.

This page intentionally left blank

Information technology – Open Systems Interconnection – Virtual Terminal Basic Class Service

1 Scope

This International Standard defines, in an abstract way, the externally visible Basic Class Virtual Terminal Service within the OSI Application Layer in terms of

- a) a model defining the interaction between users of the service;
- b) the primitive actions and events of the service;
- c) the parameter data associated with each primitive action and event;
- d) the relationship between, and the valid sequences of, these actions and events.

The service defined in this International Standard is that which is provided by the OSI Basic Class Virtual Terminal Protocol (in conjunction with the Association Control Service Element and the Presentation Service) and which may be used by any user including other Application Service Elements. The relationship between the standards for Virtual Terminal Service, Virtual Terminal Protocol, ACSE, Presentation Layer Service and the user of the Virtual Terminal Service is shown in figure 1.

This International Standard also defines two standard default virtual terminal environment profiles and describes the form of registered virtual terminal environment profiles and control objects. Virtual terminal environment profiles define sets of virtual terminal environment parameters for use in the establishment of virtual terminal associations and subsequent negotiation. This International Standard also defines a structure

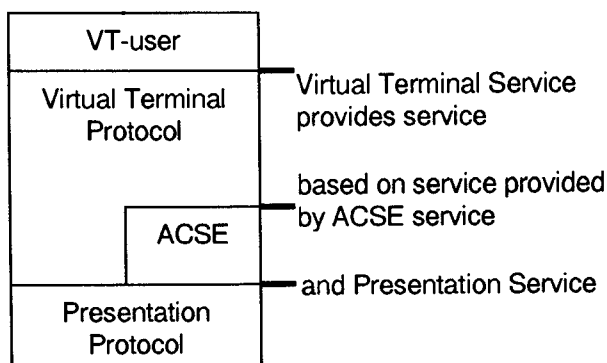


Figure 1 - Relationship of this International Standard to other OSI Application Layer Standards

of ASN.1 Object Identifiers for the objects defined in this International Standard and for use in a register of virtual terminal objects.

This International Standard does not specify individual implementations or products, nor does it constrain the implementation of entities and interfaces within a computer system. There is, therefore, no requirement for conformance to this International Standard.

This International Standard applies to interactive applications requiring terminal oriented communication expressed in terms of the transmission and manipulation of graphical images having the following characteristics:

- e) the images are composed of character-box graphic elements organised into a one, two or three dimensional structure;
- f) attributes may be associated with any graphic element to qualify its mode of display.

Control information for the communication can be modelled using virtual terminal control objects, and multiple devices can be modelled using virtual terminal device objects linked to the other virtual terminal objects.

2 Normative references

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

ISO/IEC 646:1991, *Information technology – ISO 7-bit coded character set for information interchange*.

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

ISO/IEC 2375:1985, *Data processing – Procedure for registration of escape sequences*.

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

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

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

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

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

ISO/IEC 9041-1:1997, *Information technology – Open Systems Interconnection – Virtual Terminal Basic Class Protocol – Part 1: Specification*.

ISO/IEC 9834-4:1991, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities – Part 4: Register of VTE Profiles*.

ISO/IEC 9834-5:1991, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities – Part 5: Register of VT Control Object Definitions*.

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 International Standard is based on the concepts developed in ISO/IEC 7498-1 and makes use of the following terms defined in it:

- a) application entity;
- b) Application Layer;
- c) service data unit;
- d) service access point.

It also makes use of the following terms defined in ISO/IEC 10731:

- e) service primitive;
- f) service provider;
- g) primitive;
- h) request (primitive);
- i) indication (primitive);
- j) response (primitive);

- k) confirm (primitive);
- l) confirmed service;
- m) non-confirmed service;
- n) provider-initiated service.

3.2 Association Control Service Element definitions

This International Standard makes use of the following terms defined in ISO/IEC 8649:

- a) application association;
- b) application entity title;
- c) application control service element (ACSE).

3.3 Virtual Terminal Service definitions

For the purposes of this International Standard, the following definitions apply:

3.3.1 VT-user: A user of the Virtual Terminal Service.

3.3.2 Application VT-user: The unique VT-user which can update the FDCO; if either VT-user can update this control object then neither VT-user has this designation.

3.3.3 Terminal VT-user: If one VT-user has the designation Application VT-user then the peer VT-user has the designation Terminal VT-user.

3.3.4 character-repertoire: A set of objects which can be represented by primary attribute values; one such object, represented by its primary attribute value, can occupy an array element in a display object when the character-repertoire is in use for that array element. A control object of character-string category also has an associated repertoire.

3.3.5 character-box graphic element: An atomic element of a character-repertoire where use of the repertoire has been agreed through negotiation by the VT-users.

3.3.6 primary attribute: The attribute of an array element of a display object which is a coded representation of the character-box graphic element assigned to that array element.

3.3.7 secondary attribute: The secondary attributes of an array element comprise the character-repertoire, see 3.3.4, and the rendition attributes.

3.3.8 rendition attributes: Those secondary attributes of an array element which qualify the character-box graphic element and provide information specifying how it is intended to be presented.

3.3.9 explicit modal default: The value for a secondary attribute, defined in a VTE, which is used by the text operation to update an array element if no other value is provided or already present; may also be used by the erase operation.

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

3.3.10 display object: An abstract object, defined in this International Standard, for modelling the exchange of graphic information. It consists of a number of components, see 13.1.

3.3.11 array element: That part of a display object which can hold one character-box graphic element including its primary and secondary attribute values.

3.3.12 primitive display pointer: A set of one to three coordinate values which identify a particular element in a display object.

3.3.13 extended display pointer: A set of two to four coordinate values which identify a particular array element in a block defined on a display object.

3.3.14 display pointer: Used to refer to either the primitive display pointer or the extended display pointer; whether or not blocks are in use determines which is implied.

3.3.15 logical pointer: A set of two or three coordinate values which identify a particular array element in a field defined on a display object.

NOTE – The primitive display pointer and extended display pointer do not both exist simultaneously. However, when a logical pointer exists, it is in addition to either a display pointer or an extended display pointer.

3.3.16 control object: An abstract object, of a type defined in generic terms in this International Standard, for modelling the exchange of unstructured information.

NOTE – The primary application of a control object is for modelling the exchange of information of a control nature, as understood by the VT-users; the VT Service does not constrain the interpretation of this information.

3.3.17 device object: An abstract object used to model certain logical characteristics of real devices, and to link the various objects of a virtual terminal environment together and/or to real devices.

3.3.18 object updating device: A real device capable of generating values which (possibly after undergoing a transformation) are used by one of the peer VT-users to update either a display object or control object (or possibly both).

3.3.19 VT-association: An application association between two peer VT-users.

3.3.20 VT-environment (VTE): A set of parameters that together define the data structuring and operational characteristics for a particular VT-association. The VTE exists only during the lifetime of that VT-association. The parameters of the set are mutually related by a directed graph structure. The VTE may be modified during the existence of the VT-association by negotiation.

3.3.21 current-VTE: The single VTE which exists during the Data Handling phase or the Negotiation Quiescent phase; in the Data Handling phase it is a full-VTE whereas in the Negotiation Quiescent phase it is not a full-VTE.

3.3.22 draft-VTE: The VTE, if any, under negotiation. During negotiation, the draft-VTE is not necessarily a full-VTE.

3.3.23 VTE-parameter: An individual parameter of a VTE. Each VTE-parameter is given a unique name in the service which is used as the identifier for the VTE-parameter.

3.3.24 full-VTE: A VTE that is a complete directed graph of VTE-parameters in which all node parameters and terminal leaf parameters implied by all existing nodes from the root of the tree have values.

3.3.25 VT-context-value: A collective term for the set of object instances, their assigned values and the current-VTE for a particular VT-association. A VT-context-value exists only during the lifetime of the VT-association and is normally changing continuously during this time interval.

3.3.26 reset-context: The VT-context-value which will result after a VT-BREAK service. This context value is the context after the last successful current-VTE establishment; all objects will have their initial values. If no full-VTE has been established, there is no reset-context.

3.3.27 WAVAR access-right: An access-right which can be held by at most one VT-user at any time. It is used to ensure that control and display objects cannot be updated by both VT-users simultaneously.

3.3.28 access-rule: A characteristic defined for an object in a VTE which determines which VT-users can update the object at a particular time.

3.3.29 net-effecting: The conversion of a sequence of items, representing the content of one or more update operations (see 24.3), into a different, usually shorter sequence, which results in the same final states of the objects being updated.

3.3.30 concatenation: The connection of a sequence of queued update items (see 24.3) to form a single, new, queued update item.

3.3.31 segmentation: The division of a single, queued update item (see 24.3) into a sequence of new, queued update items.

3.3.32 A-mode (Asynchronous mode): A mode of operation using two display objects, one of which is updatable by the VT-user which initiated the VT-association and the other by the peer VT-user.

3.3.33 S-mode (Synchronous mode): A mode of operation using one two-way-alternate dialogue supporting one display object; at any time, the display object may only be updated by the single VT-user which owns the WAVAR access-right.

3.3.34 service: A distinct part of the total VT Service that is composed of a sequence of primitives taken from the set {request primitive, indication primitive, response primitive, confirm primitive}.

3.3.35 sequenced service: A Service for which an indication (or confirm) primitive resulting from a corresponding request (or response) primitive is initiated in sequence with all previously initiated sequenced indications (or confirms) and their corresponding requests (or responses).

3.3.36 non-sequenced service: A Service for which an indication (or confirm) primitive resulting from a corresponding request (or response) primitive is not necessarily initiated in sequence with all previously initiated indications (or confirmations) and their corresponding requests (or responses).

3.3.37 conditionally sequenced service: A Service for which

- a) certain values for parameters of the service primitives result in sequenced operation, and
- b) other values for parameters of the service primitives result in non-sequenced operation.

3.3.38 destructive service: A service that may cause the loss of information conveyed in previously initiated services without notification of this loss to either VT-user. Only non-sequenced services may be destructive, but not all non-sequenced services are destructive.

3.3.39 non-destructive service: A service that does not cause the loss of information conveyed in previously initiated services without notification to the VT-users.

3.3.40 service parameter: A parameter defined as part of a primitive within a VT service.

3.3.41 update-window: A mechanism associated with display object addressing which defines a range of coordinate values for an array below which an update operation cannot be performed. The absolute coordinate values in the range may increase in value during operation, but cannot decrease.

3.3.42 update-window-size: A positive, non-zero integer that defines the number of contiguous array elements within an update-window.

3.3.43 trigger: Where a control object has the trigger characteristic, any update to that control object causes delivery of queued updates and, in S-mode, transfers the WAVAR access-right to the peer VT-user.

3.3.44 VTE-profile: A pre-defined set of VTE-parameter values making up a VTE.

NOTE – Some VTE-profiles are parameterised such that values for profile arguments must be supplied by the VT-users.

3.3.45 default VTE-profile: A VTE-profile, defined in annex A, that is used to establish a VTE in the absence of a VT-user specified VTE-profile at VT-association establishment.

3.3.46 registered VTE-profile: A VTE-profile registered in a register of VT Objects administered by a Registration Authority established as defined in ISO/IEC 9834-4. It has assigned to it a unique ASN.1 OBJECT IDENTIFIER value.

3.3.47 registered control object: A control object registered in a register of VT Objects administered by a Registration Authority established as defined in ISO/IEC 9834-5. It has assigned to it a unique ASN.1 OBJECT IDENTIFIER value.

3.3.48 privately defined VTE-profile: A VTE-profile whose use has been agreed privately by means outside the scope of this International Standard; it is neither a default VTE-profile nor a registered VTE-profile.

3.3.49 VTE-profile argument: An argument of a parameterised VTE-profile which must be given a value to enable a full-VTE to be made from the VTE-profile. It can be a normal VTE-profile argument or a special VTE-profile argument. Each registered VTE-profile defines the applicable arguments and their semantics.

3.3.50 normal VTE-profile argument: A VTE-profile argument which corresponds exactly to a VTE-parameter and assumes the same identifier. A VTE-profile may also specify that other VTE-parameters are also defined by the value of such an argument.

3.3.51 special VTE-profile argument: A VTE-profile argument which does not correspond to a VTE-parameter (i.e., is not from the directed graph and has no standard identifier) but has a valid use as defined in the VTE-profile definition.

3.3.52 multiple interaction negotiation (MIN): A process which enables a draft-VTE to be modified or extended in stages to create a new full-VTE acceptable to the service provider and both VT-users.

3.3.53 MIN-initiator: The VT-user that initiated the VT-START-NEG service which was successful in causing the transition to Negotiation Active phase.

3.3.54 MIN-acceptor: The VT-user that accepted the VT-START-NEG service which was successful in causing the transition to Negotiation Active phase.

3.3.55 block: A structure element of the display object. A block is a rectangular area which supports relative addressing (relative to the origin of the block) for a sub-area within a Y-array.

3.3.56 field: A structure element of the display object. A field provides one-dimensional logical addressing over a designated subset of the array elements of a Y-array of a display object.

3.3.57 field-element: A rectangular area of a Y-array of a display object forming part of a field.

3.3.58 Field Definition Control Object (FDCO): A type of control object which holds the definition of fields for a display object as Field Definition Records.

3.3.59 Field Definition Record (FDR): Holds the status and definition of one field.

3.3.60 Field Entry Instruction Control Object (FEICO): A type of control object which holds data entry instructions as Field Entry Instruction Records.

3.3.61 Field Entry Pilot Control Object (FEPCO): A type of control object which holds data entry pilots as Field Entry Pilot Records.

3.3.62 Field Entry Instruction Record (FEIR): Holds a set of data entry rules as Field Entry Instructions (FEIs).

3.3.63 Field Entry Pilot Record (FEPR): Holds a data entry pilot consisting of field entry event, field entry conditions and a sequence of field entry reactions.

3.3.64 Field Entry Instruction (FEI): Applies a rule for controlled data entry into a field.

3.3.65 Field Entry Event (FEE): An event which may occur during controlled data entry and cause an entry reaction.

3.3.66 Field Entry Condition (FEC): A condition of the data entry which taken with an entry event predicates an entry reaction.

3.3.67 Field Entry Reaction (FER): A reaction to an entry event and entry condition defined in terms of operations on other objects in the virtual terminal service.

3.3.68 Transmission Policy Control Object (TPCO): A type of control object which holds variable values which determine how field contents are transmitted after a controlled data entry.

3.3.69 Context Control Object (CCO): A type of control object which provides the local context of the VT-user at certain stages of the data entry dialogue.

3.3.70 Reference Information Object (RIO): A container for information, separate from any display object, which can be referred to during a communication.

3.3.71 Termination Conditions Control Object (TCCO): A type of control object which holds termination conditions effective on the operation of one or more device objects linked to it.

3.3.72 termination-event: A locally defined atomic event related to input from an object updating device (for example, entering a character, pressing a function key, setting a flag) which the VT-users agree (through negotiation) to designate as causing input data to be delivered to the peer VT-user.

3.3.73 ripple: Ripple is a mechanism whereby the contents of array elements of the Display Object may be moved into adjacent array elements without the need of reconveying the contents from one VT-user to the other.

3.3.74 Ripple Mode Control Object (RMCO): a type of control object associated with the ripple mechanism which controls the operation of the ripple, for example, the extent of the ripple.

3.3.75 ripple-extent: A ripple-extent is that part of a DO within which the ripple mechanism operates.

3.3.76 copy buffer: A copy buffer stores the content and structure of an extent of the Display Object to allow subsequent transfer of this structure and content to some other extent of the DO.

3.3.77 extended-y-array: The set of array elements which are within the currently defined blocks of a particular y-array.

3.3.78 extended-z-array: The set of array elements which are within the currently defined blocks of some y-array of the z-array.

3.3.79 filling: Is the operation which defines the values of the primary and secondary attributes of character box elements which are left undefined as the result of ripple operations.

3.3.80 ripple coordinate: Determines the coordinate direction of the ripple, i.e., the coordinate which is altered as the result of ripple operations, and hence determines the unit of ripple. It takes a value from the set ("x", "y", "z", "k").

3.3.81 ripple direction: Specifies whether units are to be moved in a forward (increasing coordinate) or a backward (decreasing coordinate) direction, in a ripple operation.

3.3.82 unit of ripple: Determines the units which the ripple operations work upon. It takes a value from the set ("array element", "x-array", "y-array").

4 Abbreviations

4.1 General

ACSE	Association Control Service Element
ASN.1	Abstract Syntax Notation One
BNF	Backus-Naur Form
ECMA	European Computer Manufacturers Association
IRV	International Reference Version (of ISO/IEC 646)
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
MIN	Multiple Interaction Negotiation
OSI	Open Systems Interconnection
QOS	Quality of Service
VT	Virtual Terminal
VTE	Virtual Terminal Environment

4.2 Modes of operation

A-Mode	Asynchronous Mode
S-Mode	Synchronous Mode

4.3 VTE model components

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
DSD	Data Structure Definition
DO	Display Object
ECO	Echo Control Object
FDCO	Field Definition Control Object
FDR	Field Definition Record
FEC	Field Entry Condition
FEE	Field 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
RIO	Reference Information Object
RMCO	Ripple Mode Control Object
TCCO	Termination Conditions Control Object
TCO	Termination Control Object
TPCO	Transmission Policy Control Object

4.4 Access-rules

NSAC	Not Subject to Access Control
WACA	Write Access Connection (VT-association) Acceptor
WACI	Write Access Connection (VT-association) Initiator
WAVAR	Write Access VArIable

5 Conventions

This International Standard uses the descriptive conventions contained in the ISO/IEC Service Conventions (ISO/IEC 10731).

Additional conventions used are

- a) a character string enclosed in angle brackets, i.e. <xxx>, denotes the equivalent ISO/IEC 646 mnemonic;
- b) a character string of the form n...N, where n is usually 0 or 1, denotes the set of integers greater than or equal to n, i.e., no limit is defined in this International Standard;
- c) an informal notation analogous to ASN.1 is used in places, with apparently similar terms; where exact ASN.1 notation or terminology is intended, the usage will be introduced by reference to ASN.1;
- d) a syntax derived from Backus-Naur Form (BNF) is used for VTE-profile definitions; this is explained in annex A.

6 General features

Clauses 6 and 7 give an Overview of the Virtual Terminal Service defined in this International Standard.

6.1 Introduction

The Basic Class Virtual Terminal Service supports the interactive transfer and manipulation of graphic data by VT-users. This graphic data is structured in a manner which models the class of character-box oriented terminals. The basic structuring of graphic elements is limited to images consisting of character-box graphic elements arranged in a one, two or three dimensional array. Optional functional units provide additional structuring capabilities. Two modes of operation are defined for Basic Class, S-mode and A-mode.

6.2 Features of the Virtual Terminal Basic Class Service

The Virtual Terminal Basic Class Service offers the following services to the VT-user:

- a) the means to establish a VT-association between two peer VT-users for the purpose of enabling virtual terminal information exchange;
- b) the means to negotiate the VT functional units required;
- c) the means to negotiate a consistent set of VTE-parameters;
- d) the means to transfer and manipulate structured data in a way that is independent of the local representation of information used by each VT-user and that is independent of the way in which supporting communications media are used;
- e) the means to control the integrity of the communication;
- f) the means to terminate the VT-association either unilaterally or by mutual agreement;
- g) the means to support either synchronous (S-mode) or asynchronous (A-mode) operation between the VT-users;
- h) the means to exchange priority information to gain the immediate attention of a VT-user;
- i) the means to terminate information transfer destructively and resynchronize the activity of the VT provider;
- j) a facility for defining blocks in a display object [Blocks functional unit];
- k) a facility for defining fields in a display object [Fields functional unit, also uses feature in n)];
- l) additional optional access-rules for control objects in S-mode [Enhanced Access-rules functional unit];
- m) means to control the asymmetry inherent in typical use of these features [uses the feature in l)];
- n) a facility for defining control objects with content consisting of multiple data elements or a single partially-updateable structured data element [Structured Control Objects functional unit];
- o) a facility for controlling data entry to fields using new standard types of control object [uses the feature in n)];
- p) a facility for storing and using update information in Reference Information Objects [RIOs functional unit];
- q) a facility for establishing a VT-association with the capability to switch between the modes of operation when the VTE is changed;
- r) display object update with ripple;
- s) the reporting of exception conditions by the VT-service-provider;
- t) the selective retention of VT-context between negotiation of successive VTEs.

6.3 VT Environment (VTE) and VTE-parameters

The transfer and manipulation of graphic data takes place within a VT-environment (VTE) defined by a logically consistent set of VTE-parameter values. Certain VTE-parameters are related in that a value for one VTE-parameter can constrain the existence of or permissible values for another VTE-parameter. These relationships are partly expressed by a directed graph for the VTE-parameters and partly by explicit definition.

One and only one full-VTE exists during data transfer. A full-VTE may be modified or replaced if negotiation facilities are available.

6.4 Virtual Terminal Environment Profiles

The Basic Class VT Service provides for the use of virtual terminal environment profiles (VTE-profiles) which are sets of VTE-parameters and VTE-parameter values for use in the negotiation of the VTE (see clause 11 and annex A).

6.5 Dialogue Control

The integrity of VT communication is maintained by Dialogue Control service facilities (by applying access-rules, delivery control and delivery acknowledgement, see clauses 9 and 24), and service primitive sequencing rules which provide integrity by detecting and resolving collisions (see clause 27).

7 Communication facilities

7.1 Establishment facility

The Establishment facility provides a service that establishes a VT-association and an initial VTE for that VT-association when a VT-user first invokes the Basic Class VT Service.

7.2 Termination facility

The Termination facility provides services which enable a VT-user to terminate a VT-association either in an orderly and non-destructive manner or in an immediate and potentially destructive manner. It also provides a service which enables the VT service provider to notify the VT-users when a VT-association is terminated in a potentially destructive way.

7.3 Negotiation facility

The Negotiation facility provides services which enable peer VT-users to select, modify and replace the current-VTE.

An initial VTE is established during VT-association establishment based on the VTE-profile specified. If the VT-user does not specify a VTE-profile during VT-association establishment, a default VTE-profile is used to establish the initial VTE. This VTE may subsequently be modified or replaced depending on the type of negotiation facilities available.

The type of negotiation facility available to the VT-users is determined by the VT functional units selected during VT-association establishment. In addition to the VTE-profile selection facility available during VT-association establishment, there are two other types of negotiation, i.e., switch profile negotiation and multiple interaction negotiation.

7.4 Data Transfer facility

The Data Transfer facility provides a service which enables a VT-user to update the contents of display and/or control objects to which the VT-user is currently permitted update access.

7.5 Delivery Control facility

The Delivery Control facility provides services which enable a VT-user to control, synchronise and optionally request acknowledgement of, the release to the peer VT-user of updates entered previously using the Data Transfer facility. This includes the (negotiable) ability to "quarantine" (i.e., hold back) such updates from such release until the release is requested.

7.6 Dialogue Management facility

In S-mode, the Dialogue Management facility enables the VT-users to request or cede ownership of the WAVAR access-right.

In A-mode, the Dialogue Management facility is not available.

7.7 Interrupt facilities

The Interrupt facilities are available in both modes of operation but can only be used in the Data Handling phase. There is a destructive interrupt facility and a non-destructive interrupt facility.

The **destructive interrupt** facility allows a VT-user to interrupt a previously initiated sequence of updates to display and control objects, discard all updates currently being exchanged and resume exchanging updates after the VT-service-providers have resynchronized their activities.

The **non-destructive interrupt** facility allows VT-users to exchange priority information but without destroying non-priority information; it can be used as an "attention" signalling mechanism (via appropriately defined control objects, see clause 14).

7.8 Exception Reporting facility

The Exception Reporting facility provides services which allow the VT-service-provider to report abnormal conditions to the VT-users without causing termination of the VT-association.

8 Modes of operation

Clauses 8, 9, 10 and 11 define general aspects of the Virtual Terminal Service.

The Virtual Terminal Basic Class Service supports two modes of operation. Each mode is characterised by its form of dialogue control. The initial mode of operation is selected when the VT-association is established. The ability to change modes on the establishment of a new VTE is also selected when the VT-association is established.

8.1 S-mode

S-mode (synchronous mode) has the following characteristics:

- one or other VT-user may own the WAVAR access-right; ownership may be passed between the two VT-users;
- only a single display object is supported with a single two-way-alternate dialogue. Update access to this display object is controlled by the WAVAR access-rule, see clause 9;
- the VT-users may define and make use of termination condition parameters in device objects;
- the access-rules available for any control object are as in clause 9, table 1.

8.2 A-mode

A-mode (asynchronous mode) has the following characteristics:

- the WAVAR access-right is not available;
- two display objects are supported, each with a monologue. One display object has access-rule WACI and can only be updated by the VT-user which initiated the VT-association; the other display object has access-rule WACA and can only be updated by the VT-user which accepted the VT-association;
- the VT-users may define and make use of echo control objects and termination condition parameters in device objects;
- the access-rules available for any control object are as in clause 9, table 1;
- Dialogue Management service primitives do not apply.

9 Access-rules

Each display object is assigned an access-rule which determines whether and when each VT-user is permitted to update that display object. Each control object is assigned an access-rule or combination of access-rules which determine whether and when each VT-user is permitted to update that control object. The possible access-rules are:

- **no-access** : neither VT-user may update the object;
- **WACI** : the object may be updated only by the VT-user which initiated the VT-association;

- **WACA** : the object may be updated only by the VT-user which accepted the VT-association;
- **WAVAR** : the object may be updated only by the VT-user owning the WAVAR access-right;
- **WAVAR & WACI** : the object may be updated only by the VT-user which initiated the VT-association and only when it owns the WAVAR access-right;
- **WAVAR & WACA** : the object may be updated only by the VT-user which accepted the VT-association and only when it owns the WAVAR access-right;
- **NSAC** : the object may be updated by either VT-user at any time.

In S-mode, the single display object always has access-rule WAVAR. In A-mode, one display object has access-rule WACI and the other has access-rule WACA.

NOTES

1 A display object cannot have an access-rule which is a combination of multiple access rights. However, other special access conditions can apply to a display object, e.g., due to data entry conditions applied by special control objects.

2 Use of the extended access-rules given by the Enhanced Access-rules functional unit is independent of actual use of the other functional units although certain of the other functional units may require this functional unit to be present.

The access-rules which may be assigned to control objects in each mode are listed in table 1.

In S-mode, most services may only be initiated when the WAVAR access-right is held, see 27.4. Communication facilities are available which allow a VT-user to request or cede ownership of the WAVAR access-right, see clause 33.

Table 1 – VT Access-rule and Mode correspondence with Enhanced Access-rules functional unit

Access-rules	S-mode	A-mode
no-access	(1) YES	(1) YES
WACI	(1) YES	YES (2)
WACA	(1) YES	YES (2)
WAVAR	YES (2)	NO
WAVAR&WACI	(1) YES	NO
WAVAR&WACA	(1) YES	NO
NSAC	YES	YES

NOTES

- 1 Available with Enhanced Access-rules functional unit.
- 2 Display object can only have a value from this set.

10 VT functional units

The Basic Class Virtual Terminal Service provides a number of optional capabilities called Functional Units. The VT functional units required are selected during VT association establishment.

The optional functional units available in the VT service are:

- a) Switch Profile Negotiation;
- b) Multiple Interaction Negotiation;
- c) Negotiated Release;
- d) Urgent Data;
- e) Break;
- f) Enhanced Access-rules;
- g) Structured Control Objects;
- h) Blocks;
- i) Fields;
- j) Reference Information Objects;
- k) Ripple;
- l) Exceptions;
- m) Context Retention.

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

Although formally independent, use of the Fields functional unit requires a control object which requires the Structured Control Objects functional unit.

The Exceptions functional unit can be selected only if the Break functional unit is selected.

All provisions in this International Standard which are not part of one of the optional functional units are part of the Kernel functional unit and are always available.

10.1 Negotiation functional units

The Virtual Terminal Basic Class Service provides negotiation services to enable the creation and modification of the single VTE in a manner acceptable to the service provider and both VT-users. Negotiation is available as part of the establishment facility. Additional negotiation services are optionally available (dependent on the functional units selected during association establishment) which provide two forms of negotiation, a single interaction switch profile negotiation and a multiple interaction negotiation.

At VT-association establishment, a current-VTE is established using the VTE-profile-based negotiation function embedded in the VT-ASSOCIATE service. This current-VTE may or may not be a full-VTE. If the initiating VT-user does not specify a VTE-profile with the VT-ASSOCIATE (to supply the basis for negotiating the current-VTE), the service selects a default VTE-profile appropriate to the mode of operation, see annex A. Depending on the functional units selected, this current-VTE may or may not be subsequently modified using negotiation services. If the initial current-VTE is not a full-VTE,

it is necessary to use the negotiation facility to establish a full-VTE before the Data Handling phase may be entered.

The negotiation services do not modify the current-VTE unless the negotiation reaches a satisfactory conclusion. During an instance of negotiation, agreements on VTE-parameter values are recorded in a temporary VTE definition known as the draft-VTE. In the case of single interaction profile switch negotiation, a draft-VTE only has a transitory existence. However, for multiple interaction negotiation, the concept is more significant, i.e., the draft-VTE contains the new VTE as it is built up over an extended time interval.

10.1.1 Switch Profile Negotiation

The single interaction Switch Profile Negotiation is provided by a single confirmed service, VT-SWITCH-PROFILE. This form of negotiation is similar to that provided in VT-ASSOCIATE and takes the form of a proposal made by one VT-user using a VTE-profile and, where required, offered VTE-profile argument values. The VTE-profile chosen is not negotiable but offered VTE-profile argument values may be adjusted by both the service provider and the accepting VT-user (in that order) to obtain an agreed set of VTE-parameter values, but only within the range of freedom offered by the service initiator. Adjustment of VTE-parameter values by this service is constrained to those parameters defined using VTE-profile arguments in the VTE-profile specification.

10.1.2 Multiple Interaction Negotiation

Multiple Interaction Negotiation (MIN) is initiated and terminated by confirmed services thus ensuring the agreement of both the VT-users and the service provider to the proposed action.

MIN is provided to enable the negotiation of a set of VTE-parameter values comprising a full-VTE in a series of steps (the intermediate steps do not necessarily form a complete or consistent full-VTE).

MIN negotiation is supported by four services:

- a) INVITE: a VT-user invites the peer VT-user to propose values for VTE-parameters;
- b) OFFER: a VT-user proposes values or value ranges to the peer VT-user; this may be in response to an INVITE or as a counter-offer to a previous OFFER;
- c) ACCEPT: a VT-user accepts all or a subset of the VTE-parameter values proposed by the peer VT-user in an OFFER;
- d) REJECT: a VT-user rejects VTE-parameter values proposed by the peer VT-user in an OFFER.

Using these services, independent sequences of MIN service primitives may be used to negotiate values for varying groupings of VTE-parameters. Valid MIN sequences and the rules for use of the MIN services are defined in 30.3.

NOTE – The stages of a MIN negotiation may be independent or may depend on the results of earlier stages.

When MIN is terminated, either

- the draft-VTE replaces the original current-VTE and is available for use, or
- the draft-VTE is discarded and the situation prior to the entry to MIN is restored; this may be with no full-VTE in existence.

Definitions of service phases associated with MIN are given in clause 27.

10.2 Negotiated Release functional unit

The Negotiated Release functional unit enables a VT-user to reject a request by the peer VT-user to release a VT-association and return to the phase of the VT service in effect before the release request. If this functional unit is not selected, a release request cannot be rejected.

10.3 Urgent Data functional unit

The Urgent Data functional unit provides a capability to allow a small amount of information to be conveyed from one VT-user to its peer in an urgent manner, possibly bypassing previous information exchanges. This functional unit is used to enhance support for the non-destructive interrupt facility, see 7.7, 20.1.4 (note 2) and B.16.2.

10.4 Break functional unit

The Break functional unit supports the destructive interrupt facility, see 7.7 and clause 34.

10.5 Enhanced Access-rules functional unit

The Enhanced Access-rules functional unit extends the access-rules available for control objects. In A-mode, the rule no-access is added. In S-mode, the rules WACI, WACA, WAVAR & WACI, WAVAR & WACA and no-access are added, see table 1. If this functional unit is not selected the current-VTE cannot contain COs which use these additional values of CO-access.

Table 1 lists the available access-rules in both modes of operation with and without this functional unit, see clause 9.

10.6 Structured Control Objects functional unit

The Structured Control Objects functional unit allows a parametrically defined control object to have more than one data element where the category of each data element need not be identical. This functional unit allows independent updating of the individual data elements. It also allows a control object defined non-parametrically (either in this International Standard or in a VTE-profile or as a registered control object) to be partially updated in a way defined specifically for that control object.

If this functional unit is not selected, the current-VTE cannot contain control objects requiring the above capability.

10.7 Blocks functional unit

The Blocks functional unit provides block structuring capability for a display object in a VTE. Actual use of the capability is optional, being selected for a display object by a VTE-parameter. If this functional unit is not selected the current-VTE cannot contain display objects with this capability.

10.8 Fields functional unit

The Fields functional unit provides field structuring capability for a display object in a VTE. Actual use of the capability is optional, being selected for a display object by a VTE-parameter. If this functional unit is not selected the current-VTE cannot contain display objects with this capability.

10.9 Reference Information Objects functional unit

The Reference Information Objects functional unit allows this particular form of control object to be used in a VTE. If this functional unit is not selected, the current-VTE cannot contain Reference Information Objects.

10.10 Ripple functional unit

The Ripple functional unit provides insertion, deletion and copy operations for a display object. These operations are available in two forms, either using basic addressing or using logical addressing. If this functional unit is selected, the Structured Control Objects functional unit must also be selected (so that the RMCO, 20.3.9, can be used).

NOTE – These functions allow simple text editing to be performed with much lower communications overhead than if the functional unit were not selected. Without this facility, editing of text with insertion and deletion would require the transmission of large parts of the Display Object between the peer VT-users. With this facility, only information relating to the actual insertion and deletion operations need be transmitted. Considerable communication savings are therefore possible.

10.11 Exceptions functional unit

The Exceptions functional unit provides mechanisms by which non-fatal exception conditions may be reported by the VT-service-provider to both VT-users. A facility is provided to ensure that the integrity of the VT-context is maintained.

NOTE – Without this facility, the VT-Service-provider normally aborts the VT-association if any exception condition is discovered. With this facility, exception conditions may be classified as either fatal or non-fatal. Fatal exception conditions will still cause the VT-service-provider to issue a VT-P-ABORT primitive.

10.12 Context Retention functional unit

The Context Retention functional unit allows the retention of the information stored in selected VT objects (DOs and COs) to be retained between successive VTEs within the life time of a VT-association.

The Context Retention functional unit may be selected only if the Switch Profile Negotiation functional unit is selected.

NOTE – This functional unit will allow, for instance, the contents of a display object to be retained when a new control object is brought into existence through negotiation. Without this functional unit, such contents would be lost.

11 VT Environment Profiles (VTE-profiles)

A VTE-profile is a set of VTE-parameters with predefined values for some or all of these VTE-parameters. A VTE-profile can have a number of VTE-profile arguments which are used to provide values for any VTE-parameters in the VTE-profile which are not given pre-defined values.

Other options can be defined in the VTE-profile as being subject to special VTE-profile arguments, but a VTE-profile definition cannot extend the provisions of this International Standard (e.g., new VTE-parameters not in this International Standard cannot be defined in this way).

In order to construct a full-VTE from such a VTE-profile, explicit values must be negotiated for all VTE-profile arguments (unless the VTE-profile itself defines a default value).

VTE-profiles are used to define control object and device object semantics.

Two default VTE-profiles are defined for this service. The default VTE-profiles, one for S-mode and the other for A-mode, are defined in annex A together with a description of the notation used for defining VTE-profiles and VTE-profile arguments.

NOTES

1 VTE-profiles provide an efficient method for defining the values of a number of VTE-parameters simultaneously. They also provide a means of promoting interoperability by registering well known VTE-profiles which have been designed for a particular purpose.

2 ISO/IEC 9834-4 contains the registration procedures for the ISO Registration Authority for VTE-profiles. Widely applicable VTE-profiles will be registered and maintained by that Registration Authority. The purpose of this registry is to promote interoperability and provide ease of reference.

Other VTE-profiles may be defined for use with the VT Service. These VTE-profiles may be registered with other registration authorities as appropriate for their level of visibility. In addition, private VTE-profiles may be defined and used outside the scope of ISO.

12 The VTE Model

Clauses 12 to 24 inclusive define the VTE Model in terms of which the Virtual Terminal Service is defined.

VT-users communicate by means of a shared conceptual communication area (CCA) containing a number of abstract objects and object type definitions. Information exchange is modelled by one VT-user updating the contents of the CCA and the changed state of the CCA then being made accessible to the peer VT-user. The VT-users update the CCA via the services provided by the service provider (see figure 2).

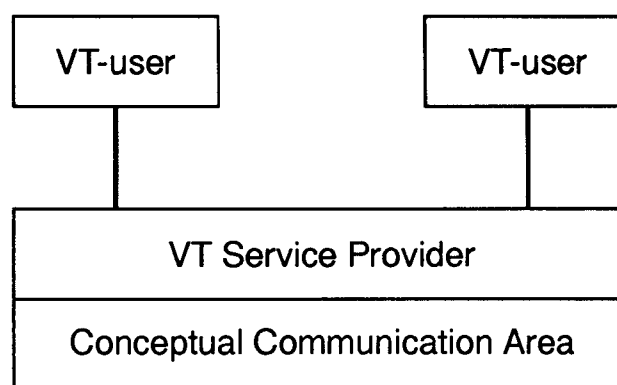


Figure 2 - Service model

The CCA, containing the abstract objects used for describing the VT service, has no physical existence.

The CCA contains the following components:

- a CDS containing one or two display objects;
- a CSS containing zero or more control objects which may be used not only for device control but also for signalling and status information and for any other purposes agreed outside the scope of the standard;
- an ACS which records which VT-user, if any, owns the WAVAR access-right;
- zero, one or more device objects each of which represents a mapping between a display object and a real device and provides VTE-parameters which enable some control over this mapping. A device object is linked to one display object and to one or more control objects; clause 16 describes the concept in more detail;
- a DSD that contains the object type definitions for the display objects, device objects, control objects (see clauses 13, 14 and 16) and other negotiated VTE-parameters which form the principal part of the current-VTE definition.

Figure 3 illustrates these components of the CCA and their relationships (i.e., linking together) and also indicates how display objects and control objects can be mapped onto real devices via related device objects.

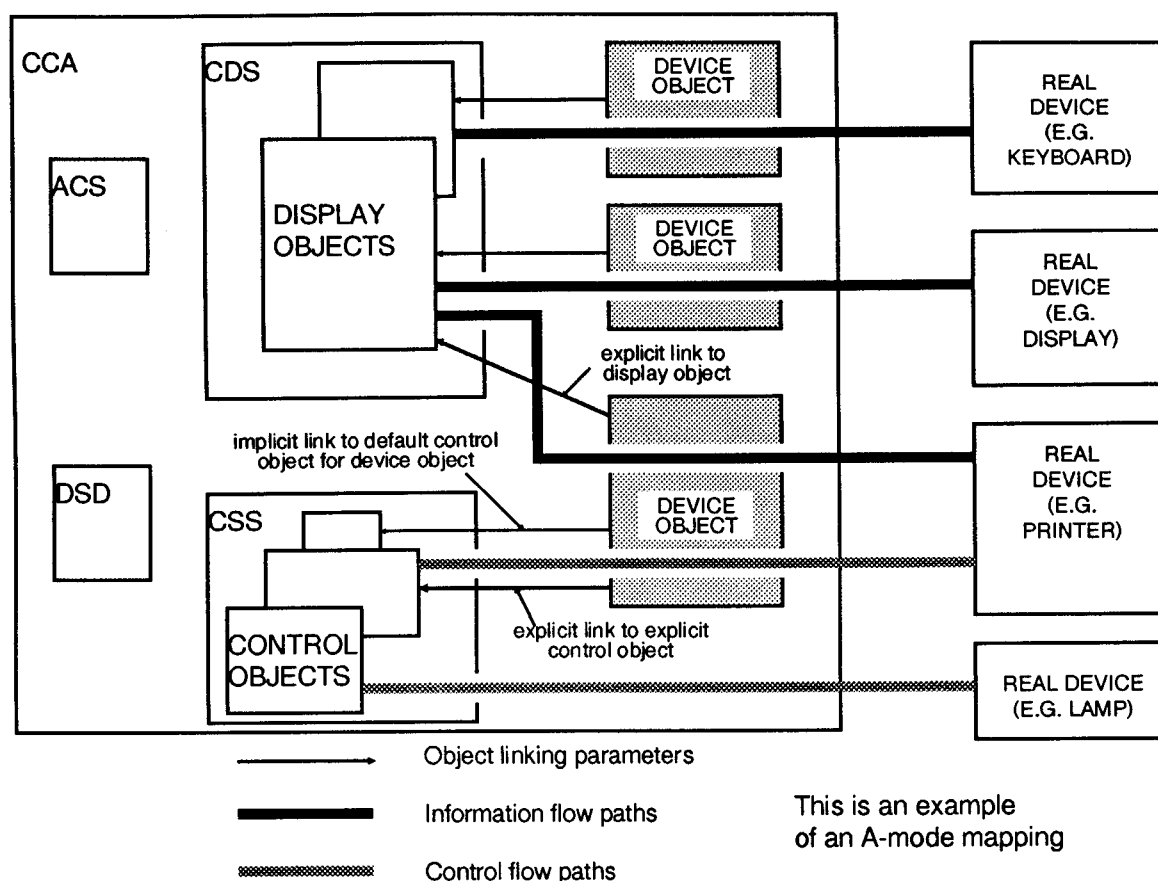


Figure 3 - CCA components, linking and mapping

The contents of the DSD are initialised to correspond to the current-VTE at VT-association establishment time and may only be changed subsequently by negotiation. The contents of the remaining sub-areas are open to change when the VT-association is in existence and negotiation is not in progress.

NOTES

1 The model described thus far is symmetrical. In actual use, however, the communication facilities provided by this model may be used in an asymmetrical manner. The method for assigning asymmetrical roles in practice uses the access-rules WACI and WACA, which are available in S-mode as well as A-mode when the Enhanced Access-rules functional unit is in use, see clauses 8 and 9.

Assignment of one of these access-rules to a RIO (clause 15) or any control object (clause 20 and clause 14) can have implicit semantics relating to the 'ownership' or 'location' of the object.

An asymmetry to the VT-users' view of the use of device objects can be assigned by means of the *device-default-CO-access* VTE-parameter (clause 23) but this asymmetry of device object use has no meaning to the VT service provider.

2 Clauses 25 onwards define the communication facilities and the primitives which VT-users use to modify the CCA. This International Standard does not preclude local mechanisms or "macro" operations which combine primitives at a local interface.

13 Display Objects

This International Standard does not define the semantics associated with the structure of a Display Object (DO), nor the relationship between this structure and the real device. Such semantics, i.e., the meaning of the update operations which are performed on the display object structure, depends on the VTE-profile used (see clause B.11).

The Kernel functional unit of the Basic Class Virtual Terminal Service defines a basic structure for the display object. This basic structure can be significantly extended if use of the Blocks or Fields functional units is selected for a particular DO, see 10.7, 10.8, 13.1.2 and 13.1.3.

Blocks are deemed to be in use for a particular DO if VTE-parameter *block-definition-capability* is "yes" for that DO. Fields are deemed to be in use for a particular DO if VTE-parameter *field-definition-capability* is "yes" for that DO (see 18.1).

13.1 Structure

A DO has the following components:

- a) a one, two or three dimensional character array of array elements, each capable of holding one character box graphic element from a repertoire of such elements (see 13.2 and 18.2.4);

- b) a display pointer, see 13.1.1.1;
- c) a set of modal attribute values, see 13.1.1.2;
- d) a set of global attribute values, see 13.1.1.3;
- e) possibly, a logical pointer, see 13.1.3.3;
- f) possibly, sets of field modal attribute values, see 13.1.3.2.

13.1.1 Character array

Each array dimension has a set of integer coordinate values 1...n so that the lower bound is 1. An upper bound may be defined or the array dimension may be declared unbounded (no limit to the value of n). There are no restrictions on combinations of bounded and unbounded dimensions.

The dimensions are named X, Y, and Z. X is the lowest order dimension. A set of array elements identified by a contiguous set of x-coordinate values 1...n (where n is the bound of X if X is bounded) at a particular value of y is known as an **X-array**. Y is the next higher order dimension. A set of contiguous y-coordinate values 1...n (where n is the bound of Y if Y is bounded) at a particular value of z identifies a set of X-arrays known as a **Y-array**. Z is the highest order dimension. A set of contiguous z-coordinate values 1...n (where n is the bound of Z if Z is bounded) identifies a set of Y-arrays known as the **Z-array** (see also clause B.11).

The dimensions determine the ordering of all the addresses of the display object in that address (x,y,z) is less than (precedes) (x',y',z') precisely if

$$\begin{array}{l} \text{either } z < z' \\ \text{or } z = z' \text{ and } y < y' \\ \text{or } z = z' \text{ and } y = y' \text{ and } x < x'. \end{array}$$

This ordering rule applies to the operations defined in clause 19.

13.1.1.1 Display Pointer

The display pointer is either a primitive display pointer or an extended display pointer. The primitive version applies when blocks are not in use. The extended version applies when blocks are in use. The **primitive display pointer** consists of a set of (one, two or three) coordinate values which identifies a particular array element or can identify a position immediately after a bounded X-array where there is no array element (see 19.1.1.1). The **extended display pointer** consists of a set of coordinate values (p,q,b,z), where q and z are only used if the y and z dimensions respectively are defined. It (usually) identifies an array element, as defined in 13.1.2. The initial value for either pointer when a VTE is first brought into use (initialized) is 1 for all applicable coordinate values.

13.1.1.2 Modal Attributes

The set of modal attributes consists of a value for each of the secondary attributes (see 13.2). Each value is either "null" or is an explicit value for the attribute from the set of values negotiated for this attribute. These values are used as defined in clause 19. The initial values, when a VTE is first brought into use (initialized), are "null" for all attributes.

13.1.1.3 Global Attributes

The set of global attributes consists of a value for each of the secondary attributes (see 13.2) except character repertoire. Each value is either "null" or is an explicit value for the attribute from the set of values negotiated for this attribute. These values are used as defined in clause 19. The initial values when a VTE is first brought into use (initialized) are "null" for all attributes.

13.1.2 Block definition

A **block** is a rectangular sub-area of a Y-array specific to that Y-array. It has an origin designated by an (x,y) coordinate value and a width in X and a depth in Y denoted as Dx, Dy respectively. Any edge of a block can coincide with any edge of the Y-array but cannot go outside the bounds of X and Y. Dx and Dy must be positive and non-zero. When the block-definition-capability is used, the X dimension and, if defined, the Y dimension must be bounded.

Figure 4 illustrates the block concept.

If the Y-dimension is not defined for the display object, the Y-array and all blocks are implicitly one X-array deep, and all references to Y dimension must be amended accordingly.

A block establishes an addressing sub-domain of the Y-array which contains some subset of the array elements of the Y-array. Within the block the contained array elements are addressed using p and q coordinate values which are interpreted as relative to the origin of the block. The p and q coordinates are subject to the addressing constraints negotiated for the X and Y dimensions respectively of the unstructured DO.

The subset of an X-array which is contained within a block is termed an X-subarray or P-array.

A number of blocks can be defined in each Y-array. The maximum number is a negotiable parameter of the DO and can be unbounded. The block structure on a Y-array is independent of that on any other Y-array. The blocks of a Y-array are addressed by a pseudo-dimension known as the B dimension, with b used to denote the coordinate value.

The address of a block is a value (b,z), the address of a X-subarray (P-array) in a block is (q,b,z) and of an array element within a block is (p,q,b,z), where q and z are used only if Y and Z dimensions respectively are defined. The address (p,q,b,z) identifies the array element (x,y,z) where

$$x = X_0 + p - 1$$

$$y = Y_0 + q - 1$$

and (X_0, Y_0) is the origin of the block (b,z).

Extended addresses, i.e., including the block coordinate, are ordered by an extension of the ordering algorithm in 13.1.1. Address (p,q,b,z) is less than (precedes) (p',q',b',z') precisely if

$$\begin{array}{l} \text{either } z < z' \\ \text{or } z = z' \text{ and } b < b' \\ \text{or } z = z' \text{ and } b = b' \text{ and } q < q' \\ \text{or } z = z' \text{ and } b = b' \text{ and } q = q' \text{ and } p < p'. \end{array}$$

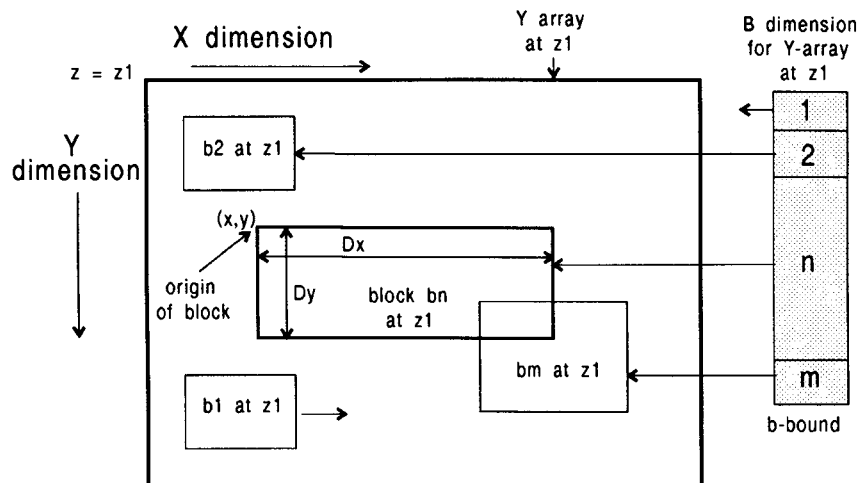


Figure 4 - Illustration of Block concept

Blocks can overlap. Thus, an array element of the Y-array can be addressed and updated by more than one block addressing mechanism, but has single values for primary and secondary attributes. See also 13.1.3.

If blocks are in use for a DO, extended addressing always applies and an array element cannot be addressed (or updated) using extended addressing unless it is contained within at least one block. See also 13.1.3.3 and 19.5.

Operations are defined in 19.1.2 and 19.4.1.5 for creating and deleting blocks and for other update operations using blocks.

NOTES

- 1 The creation or deletion of one or more blocks does not affect the content of any array element which is always that resulting from the latest update made by any mechanism, see 19.4.1.5.
- 2 In contrast to fields, see 13.1.3, no explicit storage is included in the model for block definition information.

13.1.3 Field definition

A **field** is a logical sub-area of a Y-array; it is an addressing domain with one-dimensional addressing which is independent of the actual form of the field in terms of its structure on the underlying display object. When the field-definition-capability is used, the X dimension and, if defined, the Y dimension must be bounded. Additional attribute capabilities are available with fields.

A single field is constructed of a sequence of one or more non-overlapping rectangular sub-areas of a particular Y-array known as field-elements; these are defined in a similar way to blocks but are formally distinct from blocks (they do not appear on the B dimension for the Y-array). The logical ordering of the field-elements is determined by the field definition not the position on the Y-array. See also 13.1.3.1.

A field can be of any shape within the bounds of the Y-array but may not overlap another field. A field may overlap one or more blocks. Thus an array element of the Y-array can be

addressed and updated by one field and by one or more block addressing mechanisms, but has single values for primary and secondary attributes.

A number of fields can be created on a Y-array. The maximum number is a negotiable parameter for the DO and can be unbounded. The field structure on a Y-array is independent of that on any other Y-array. The fields of a Y-array are addressed by a pseudo-dimension known as the F dimension, with f used to denote the coordinate value. Each value of f can potentially address one complete field, however many field-elements it contains (but at any particular time the field may be "non-extant", see 20.3.3.5).

The 'logical' addressing of array elements within a field is denoted by the pseudo-dimension K with k used to denote a coordinate value on this. See also 13.1.3.1.

Figure 5 illustrates the field concept.

Fields are addressed using the logical pointer, see 13.1.3.3. The address of a field is a value (f,z) and the address of an array element within a field is (k,f,z) , where z is used only if the Z dimension is defined. The relative k coordinate is interpreted such that value 1 (the minimum) addresses the origin position of the first field-element of the field in the field definition. The maximum value for k is the total number of array elements contained in all the field-elements in the field. (A field cannot be "unbounded".)

NOTE 1 – The z coordinate of the logical pointer is independent of the display pointer for the underlying display object. Thus, when switching between logical addressing and either block addressing or addressing the underlying display object, the values of the respective pointers are preserved.

Logical addresses are ordered as follows: address (k,f,z) is less than (precedes) (k',f',z') precisely if

either $z < z'$
 or $z = z'$ and $f < f'$
 or $z = z'$ and $f = f'$ and $k < k'$.

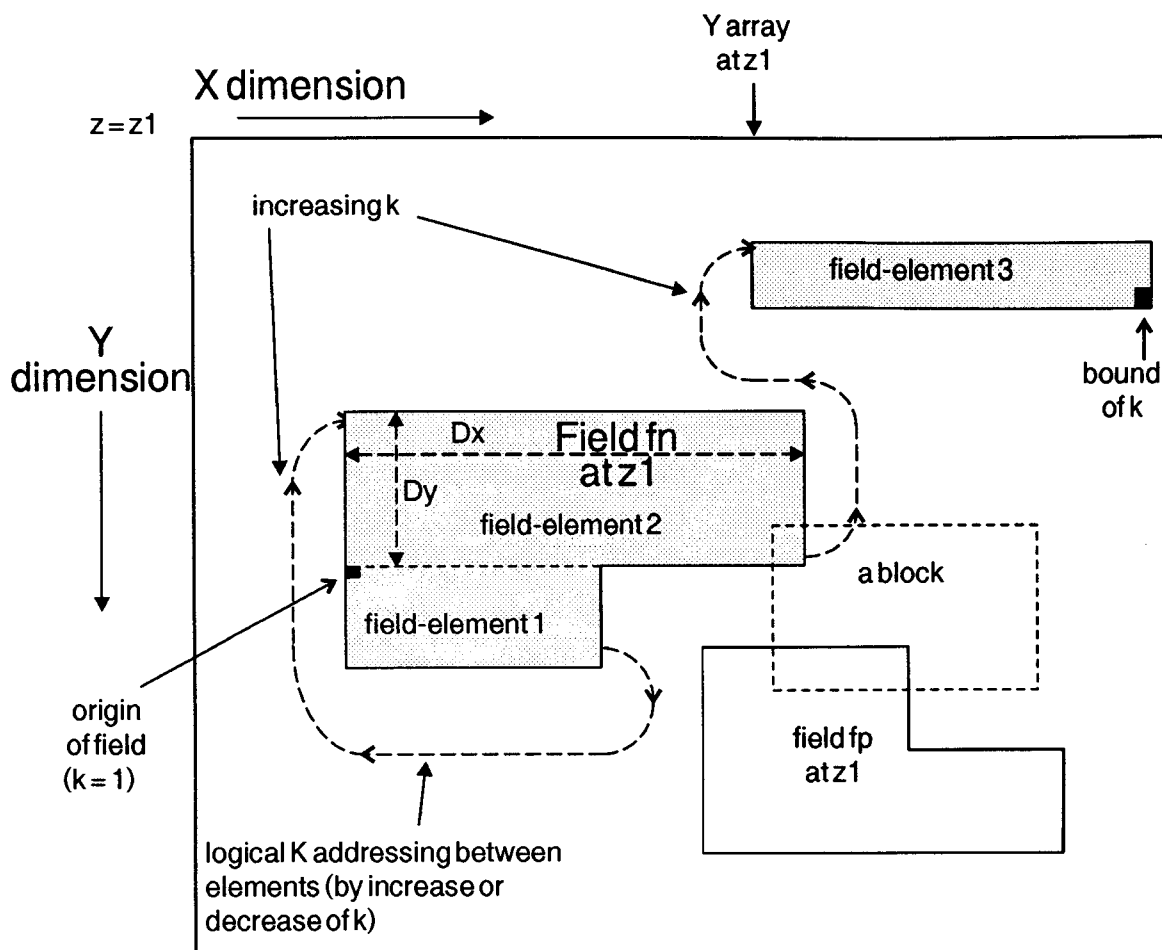


Figure 5 - Illustration of Field concept

NOTE 2 – The presence of explicit navigation path links, see 20.3.3.4, has no effect on the ordering as defined by the above algorithm and as used by operations which refer to this ordering.

The relationship between the k coordinate values and the (x,y) coordinates of the array elements contained in a field is defined in 13.1.3.1.

Field definitions are held in a FDCO associated with the DO, see 14.2 and 20.3.3. Fields are Created, Modified or Deleted by update operations on the FDCO, see 20.3.3.5. A particular (z,f) value may or may not have an associated defined field at any particular time.

Operations are defined in clause 19 for update operations using fields.

In addition to the normal access control applied by the display object access-rule, further rules for the control of updates by the terminal VT-user can be applied using the FEICO and FEPCO types of control object, see 19.5 and clause 20.

When field-definition-capability is selected for a DO, update access to the DO other than by the logical pointer (13.1.3.3) is subject to the VTE-parameter *access-outside-fields* (18.2.2); see also 19.5.

13.1.3.1 Logical addressing by K dimension

The following defines the structure of a field and defines the relationship of the logical addressing of the contained array elements by the K dimension to the (x,y) coordinates on the underlying Y-array (display pointer addressing).

The field-elements of a field are similar in shape to blocks, having a width in the X-dimension and a depth in the Y-dimension, see 13.1.3. For the purpose only of defining the K -dimension, a pseudo-dimension referred to as the S -dimension is introduced. The field-elements are arranged on this in the order in which they are defined in the field extent in the FDR (20.3.3.3), and are considered to have S coordinate values from 1 to some value S_m determined by the number of field-elements (may be 1). S coordinate values do not appear in any pointer and no means of explicitly changing s coordinate value is provided.

The K -dimension in effect links through all the field-elements of one field in increasing S coordinate value. Thus, $k=1$ corresponds to $(1,1,s=1)$ and increasing k values move along the first x -subarray to its bound $(x_m,1,s=1)$. Unless this is the last X -subarray in the field-element, an automatic next- X -subarray operation, to $(1,y:=y+1,s=1)$, occurs. Decreasing k

values will cause the reverse move. At the bound (x_m, y_m, s) of the last x-subarray in a field-element (not being the last field-element of the field), an automatic 'next-field-element' operation occurs, i.e., to $(1, 1, s:=s+1)$. Decreasing k values move from the origin $(1, 1, s)$ of one field-element (not being the first) with an automatic 'previous-field-element' operation to $(x_m, y_m, s:=s-1)$, i.e., to the maximum values of x and y for field-element s-1. The bound of K is when x and y reach their maximum values in the last field-element S_m . See also B.18.2.

13.1.3.2 Modal Attributes for Fields

When field-definition-capability is selected for a DO, there is a capability for associating with each field a set of secondary attribute values known as the **field modal attributes**. Each set consists of a value for each of the secondary attributes (see 13.2). Each value is either "null" or is an explicit value from the set of values negotiated for this attribute. These field modal attribute values are set or used by certain operations on fields as defined in clause 19.

NOTE – Although field modal attributes are not formally part of the FDRs which otherwise define the fields, it is implicit that a set of field modal attributes can be stored for every FDR which can be in an "active" or "inactive" state, see 20.3.3.

13.1.3.3 Logical Pointer

When fields are in use for a DO, an additional form of pointer is brought into use, the **logical pointer**. This has the form (k, f, z) where z is present only if Z dimension is defined. The initial value for the pointer when a VTE is first brought into use (initialized) is 1 for all applicable coordinate values.

The display pointer (in primitive or extended form) and logical pointer are independent of each other and a change to one does not affect the value of the components of the other. Thus, when changing between logical addressing and either block addressing or addressing the underlying display object, the values of the respective pointers are preserved.

NOTES

1 When fields are in use for a DO, an array element which is not in the address domain of any field (when mapped onto underlying physical coordinates) cannot be addressed by the logical pointer and thus cannot be updated by any logical operation.

2 An array element for which the condition in note 1 above is satisfied and which is not within a block cannot be addressed or updated.

13.2 Attributes

At any given time, an array element is either empty (i.e., it has no primary attribute value currently assigned to it) or it contains a **primary attribute** value selecting a character-box graphic element from a repertoire of such elements.

Each array element also has a number of **secondary attributes**:

- character-repertoire;
- font;
- emphasis;
- foreground-colour;
- background-colour.

The last four are referred to as the **rendition attributes**.

Each rendition attribute is independent of other rendition attributes and can have multiple values negotiated for its use.

The rendition attribute *font* is subservient to character-repertoire in that a separate set of font values for each repertoire can be negotiated with semantics specific to that repertoire.

The rendition attribute *emphasis* may have a substructure; the individual subattributes may be individually updatable.

If the primary attribute in an array element has a value assigned, then all secondary attributes necessarily have values assigned. If the primary attribute has no value assigned, each rendition attribute may or may not have a value, independently, and the character-repertoire secondary attribute has no value assigned.

Each array element, with its content of primary and secondary attributes, is self-contained and completely independent of all other array elements.

NOTES

1 The interpretation of any empty array element and the effect of any rendition attribute values, and in particular its physical representation on any real device, e.g. a display or printer, are not defined in this International Standard.

2 This International Standard does not further define the form of the emphasis attribute or whether any component subattributes are individually updatable. It includes VTE-parameters to enable a VTE-profile to define a form for this attribute and provide profile arguments for precise assignment in a VTE; see 18.2.6 and clause 23.

3 The additional facilities described in 13.1 do not alter the property that each array element of the display object has a single value for the primary attribute and each secondary attribute independently of the values in any other array elements (this value may be undefined). They do provide additional ways in which these values can be set.

14 Control objects

Control objects (COs) enable VT-users to handle control information related to virtual terminal functions and to real devices. This International Standard defines VTE-parameters determining the syntax of control objects, that is, their range of values and permitted update operations, see 20.1.

The semantics of a control object, i.e., the interpretation of update operations in relation to real devices, is determined by the CO-type-identifier VTE-parameter, see 20.1.1. This specifies the semantics of the CO as being defined by one of the following:

- a) within this International Standard;
- b) in a VTE-profile;
- c) as part of a registered control object;
- d) by means known to both VT-users outside the scope of this International Standard.

When a VT-association is established, the control objects defined in the VTE-profile selected are the control objects available in the initial VTE. Subsequent negotiation may

- e) replace the current VTE (thus making a new set of control objects, as defined in the new VTE-profile, available),
- f) add new control objects to the current VTE, or
- g) alter the characteristics of existing control objects.

A default control object is implicitly defined when each device object is defined. This default control object contains eight booleans and may be defined as having a trigger mechanism, see clauses 16 and B.4.

Each control object is assigned an unambiguous name within the scope of the VTE. This name is used to identify the control object when its information field is to be updated or its VTE-parameters are to be negotiated. A default control object is referenced using the name of the device object with which it is associated.

A control object may be linked to a device object (see clauses 16 and 23).

14.1 CO structure facilities and related restrictions

Two functional units are relevant to general control object capabilities, the Enhanced Access-rules functional unit and the Structured Control Objects functional unit.

14.1.1 Access-rules

A control object has an access-rule determining which VT-user may update it and when, see clause 9. The Enhanced Access-rules functional unit determines the available set of access-rules which can be assigned to control objects as defined in table 1.

A control object also has an update priority, see 20.1.4 and 24.5. The Enhanced Access-rules functional unit determines the values of VTE-parameter *CO-priority* which can be assigned, see 20.1.4 and table 11.

If the *CO-priority* value is "normal", the control object may also have the "trigger" characteristic. An update to a CO with the trigger characteristic causes delivery of queued updates and (in S-mode) transfers the WAVAR access-right to the peer VT-user.

14.1.2 CO information structure

The Structured Control Objects functional unit determines the capability available for defining structure in the information content of a CO.

In general, the structure of the information field of a control object may be defined either in terms of VTE-parameters or by reference to a separate definition which then accompanies the definition of the information field semantics. This separate definition can be contained within this International Standard or can be external to it, for example, in an international register.

When the information structure is defined parametrically it comprises a number of data elements; the number and the type and size of each data element are VTE-parameter values and thus can be changed by negotiation. Data elements thus defined may be individually updated without simultaneously updating other data elements of the same object.

The number of data elements can exceed one only if the Structured Control Objects functional unit is selected for the VT-association.

When the information structure is not defined parametrically changes to the elements of the structure cannot be negotiated; the structure definition may allow individual updating of elements. The information structure of the CO must be defined parametrically if the Structured Control Objects functional unit is not selected.

14.2 Standard control objects for fields and controlled data entry

This International Standard defines a number of standard types of control object which are used with the Fields functional unit. These are listed below with a brief overview of their use. Use of these COs also requires the Structured Control Objects functional unit to be selected, see clause 10. Clause 20 defines the VTE-parameters for these control objects and gives further details of their use. Annex B gives further explanatory material.

- a) **Field Definition Control Object FDCO**: mandatory if field-definition-capability is to be used for a display object. Fields are defined by Field Definition Records (FDRs) in the FDCO. FDRs can also link to records in FEICO and FEPCO, see below, to apply conditions for controlled data entry to the fields. The access-rule for the FDCO determines the role-assignment, if any, known to the service, see clause 12 and 19.5.2;
- b) **Field Entry Instructions Control Object FEICO**: optional, but useful only if a FDCO is present, see a); contains Field Entry Instructions (FEIs) which can be linked to a field by the FDR of the field;
- c) **Field Entry Pilots Control Object FEPCO**: mandatory if a FEICO is present, see b), otherwise optional, but useful only if a FDCO is present, see a); contains data entry pilots each consisting of a Field Entry Event (FEE), Field Entry Conditions (FECs) and a sequence of Field Entry Reactions (FERs) and which can be linked to a field by the FDR of the field;
- d) **Context Control Object CCO**: optional but useful only if a FDCO is present, see a); it enables an initial starting point for data entry to be indicated and similarly the point at which termination of data entry occurred, with the reason for termination (if a FEPCO is in use);
- e) **Transmission Policy Control Object TPCO**: optional but useful only if a FDCO is present, see a); contains general instructions relating to the notification of field updates which are used as the default for all fields but which for any field can be over-ridden by its FDR.

For a display object for which field-definition-capability is selected, exactly one FDCO must be defined, at most one CCO and TPCO can be defined, and one or more FEICOs and FEPCOs can be defined; FEPCOs are not directly associated

with FEICOs but at least one FEPCO must be present if any FEICOs are present. The various COs for a display object should be linked to it as recommended in 23.5; B.18.14 illustrates this. B.18.4 and B.18.5 give further explanation of the use of data entry COs.

14.3 Standard control object for dynamic termination conditions

This International Standard defines a standard type of CO, the **Termination Conditions Control Object TCCO**, which can hold dynamic termination conditions. It is available in either mode of operation. Its use requires the Structured Control Objects functional unit to be selected. A number of instances of the type can be used in a VTE, each with a VTE-unique *CO-name* value. The termination conditions in such a CO can be updated at any time when a VT-user has update access to it. If one or more device objects link to such a CO then all the termination VTE-parameters in these device objects become ineffective. However, the TCCO is overridden for a device object if that device object is linked to a FDCO, see 14.2 and 23.5. See clause 20 for further definition.

14.4 Standard control object for notifying termination

This International Standard defines a standard type of CO, the **Termination Control Object TCO**, which is used to notify the Application VT-user of the reason input was terminated and made available by the Terminal VT-user. It is available in either mode of operation and does not require the Structured Control Objects functional unit. It may be used in conjunction with termination events specified in either the termination VTE-parameters in a device object or termination events specified in a TCCO. See clause 20 for further definition.

14.5 Standard control object for echo control

This International Standard defines a standard type of control object, the **Echo Control Object ECO**, which may be used in A-mode to notify the remote VT-user whether echoing is on or off (echoing is the responsibility of the Terminal VT-user and this control object only provides an indication of whether the Application VT-user wants the Terminal VT-user to echo or not). This control object does not require the Structured Control Objects functional unit. See clause 20 for further definition.

14.6 Standard control object for ripple mode editing

This International Standard defines a standard type of control object, the **Ripple Mode Control Object RMCO** which is used as part of the ripple mode editing facility made available with the Ripple functional unit. For a display object for which ripple-capability is selected, exactly one RMCO must be defined. Use of this CO type requires the Structured Control Objects functional unit to be selected. Clause 20.3.9 defines the VTE-parameters for this CO and gives further details of its use.

The following information is held in a RMCO:

- a) a boolean switch to allow a VT-user to alternate between insertion and overwrite operation of DO updates;
- b) the primary and secondary attributes to be applied to empty space resulting from ripple operations;
- c) a control which governs the action to be taken in overflow situations at the end of an x-array or a y-array;
- d) a control which governs the action to be taken in overflow situations when using logical addressing;
- e) a control to determine whether insert and delete array operations are to be before or after the display pointer.

15 Reference Information Objects

A Reference Information Object (RIO) is a specific type of structured control object which contains object update information for reference while a VTE is in use. The nature of this CO type is defined in this clause. Later clauses define the VTE-parameter values for this object type and operations on such objects; these are an extension to the set of operations defined for other COs defined in this International Standard.

15.1 Structure

A RIO is a container for update information that may be referenced during VT operation. It is structured into a number of smaller accessible units called RIO-records, each designed to contain an independently addressable element of such information.

A RIO-record may contain any valid content for a VT-DATA request, i.e., an arbitrary sequence of updates to display objects and control objects (including RIO updates) and execute- or call- record operations (see 22.4.1 and 22.4.2) on the same or a different RIO; nesting is permitted but loops must not occur. A RIO-record can exist but be empty.

A RIO has an access-rule which is agreed as part of the definition of a RIO in a VTE. It conforms to the general provisions for control objects.

15.2 Generation and use of RIOs

One or more RIOs can be included in a VTE only if the Reference Information Objects functional unit is selected. This functional unit is independent of any other VT functional unit.

VTE-parameters to define a RIO in a VTE are defined in clause 21. They conform to the general CO VTE-parameters in clause 20.

The initial content of a RIO can be defined in a Profile, called into a profile from a register of predefined RIOs, possibly by a profile argument, or can be called directly from the register into a VTE during negotiation. Initial content of a RIO derived in this way is implicit in its definition and cannot be negotiated. The initial content can be empty.

The initial contents of a RIO can be modified or new contents added (using VT-DATA) only if an access-rule other than no-access is assigned to the RIO; the definition of the RIO may forbid or restrict such an assignment.

Operations to update a RIO and to make use of its contents are defined in clause 22; these are extensions to the update operations as defined for other control objects.

NOTE – a RIO-record can also be used as a copy buffer, see 19.1.4.10 and 19.4.2.5; such use is independent of the usage described above.

16 Device objects

Device objects provide VT-users with a mechanism for specifying certain real device related characteristics to aid in mapping information contained in display objects to/from real devices. Device objects also serve to relate devices to display objects and any control objects which may have been selected as part of a VTE (see clause 14) to assist in the operation of the device.

Device objects are selected in the initial VTE when establishing a VT-association when they exist in the selected VTE-profile.

Subsequent negotiation may cause a different set of device objects to be selected if a new VTE-profile is used. Negotiation may also define new device objects using VTE-parameters (see clause 23) or alter the characteristics of existing ones.

A device object has a single name which is unambiguous within the scope of the VTE.

Device objects are not updated directly using the Data Transfer facility. The VTE-parameters of a device object are intended to influence the mapping between a display object and a real device.

A default control object is associated with each device object. This control object does not use the normal control object VTE-parameters (clause 20) but has pre-defined characteristics and certain specific VTE-parameters in the device

object (see clause 23). For update purposes, this control object is referenced by the name of the device object.

17 VTE Parameters and directed graph

17.1 Directed graph of VTE-parameters

The VTE-parameters are structured as a directed graph as specified in figures 6 and 7. These figures also indicate those VTE-parameters which may have multiple occurrences.

For explanatory purposes, some sub-trees of the graph are shown with a "root" for which there is no actual parameter defined in this International Standard. These "phantom" roots are indicated by [...] and, of course, cannot be given values.

The individual parameters of the directed graph are defined in clauses 18, 20, 23 and 24 except for *mode*, *class* and *functional units* which are not VTE-parameters but are service parameters of the VT-ASSOCIATE service, i.e., *VT-mode*, *VT-class* and *VT-functional-units* respectively, (see 28.1.3).

NOTE – The directed graph does not show the dependency of certain VTE-parameters on VT functional units.

17.2 VTE consistency rules

A full-VTE is required to have values for VTE-parameters as follows:

- a) a value for any VTE-parameter whose existence is implied by a "parent" VTE-parameter at a higher node in the graph or as defined in this International Standard; this value may be supplied as a default if one is defined;
- b) values for multiple occurrence VTE-parameters whose existence is implied by a "parent" VTE-parameter at a higher node in the graph or as defined in this International Standard; a value may be supplied as a default if one is defined;
- c) excluding values for VTE-parameters whose non-existence is implied by the value of a parameter at a higher order node in the graph or as defined in this International Standard.

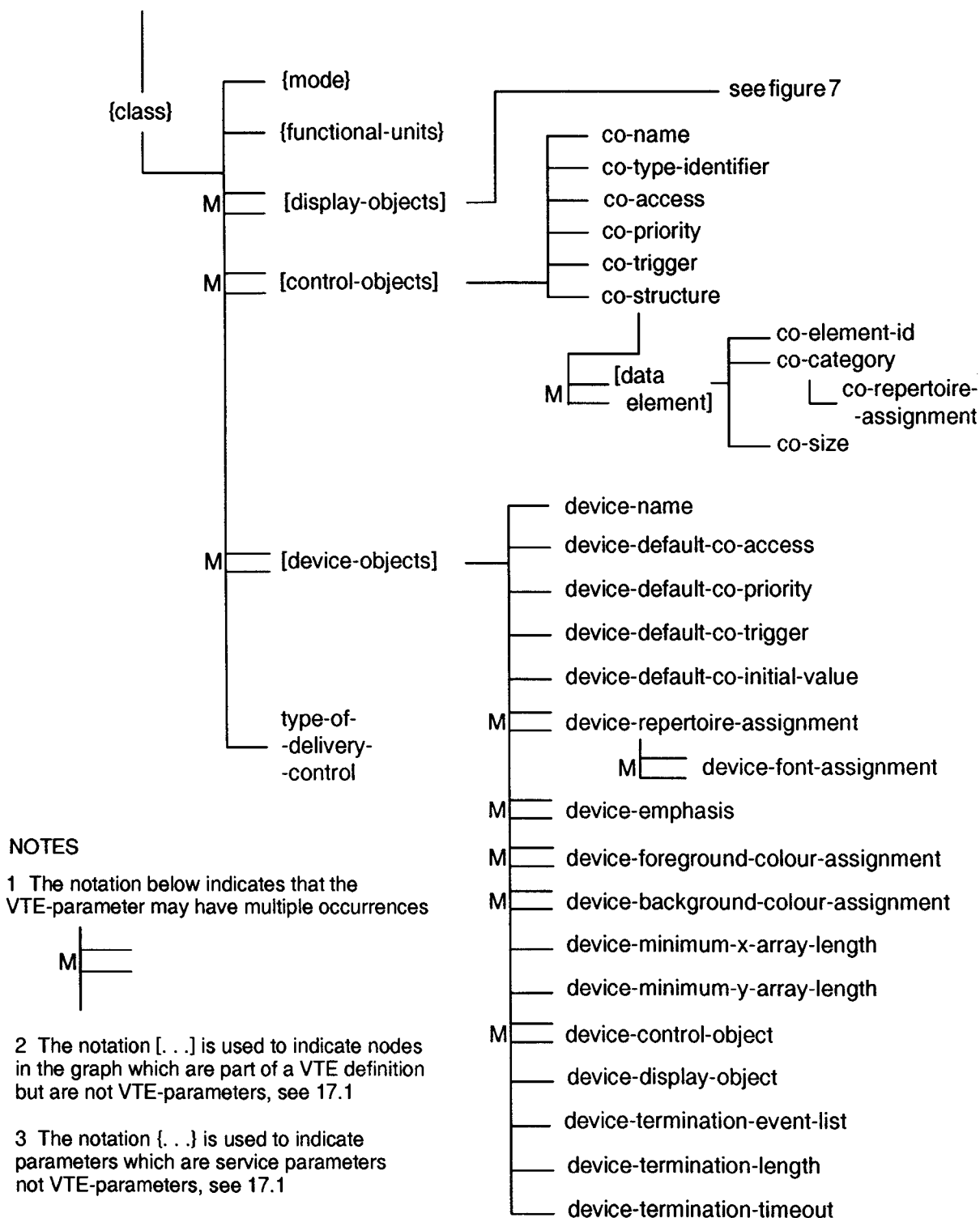
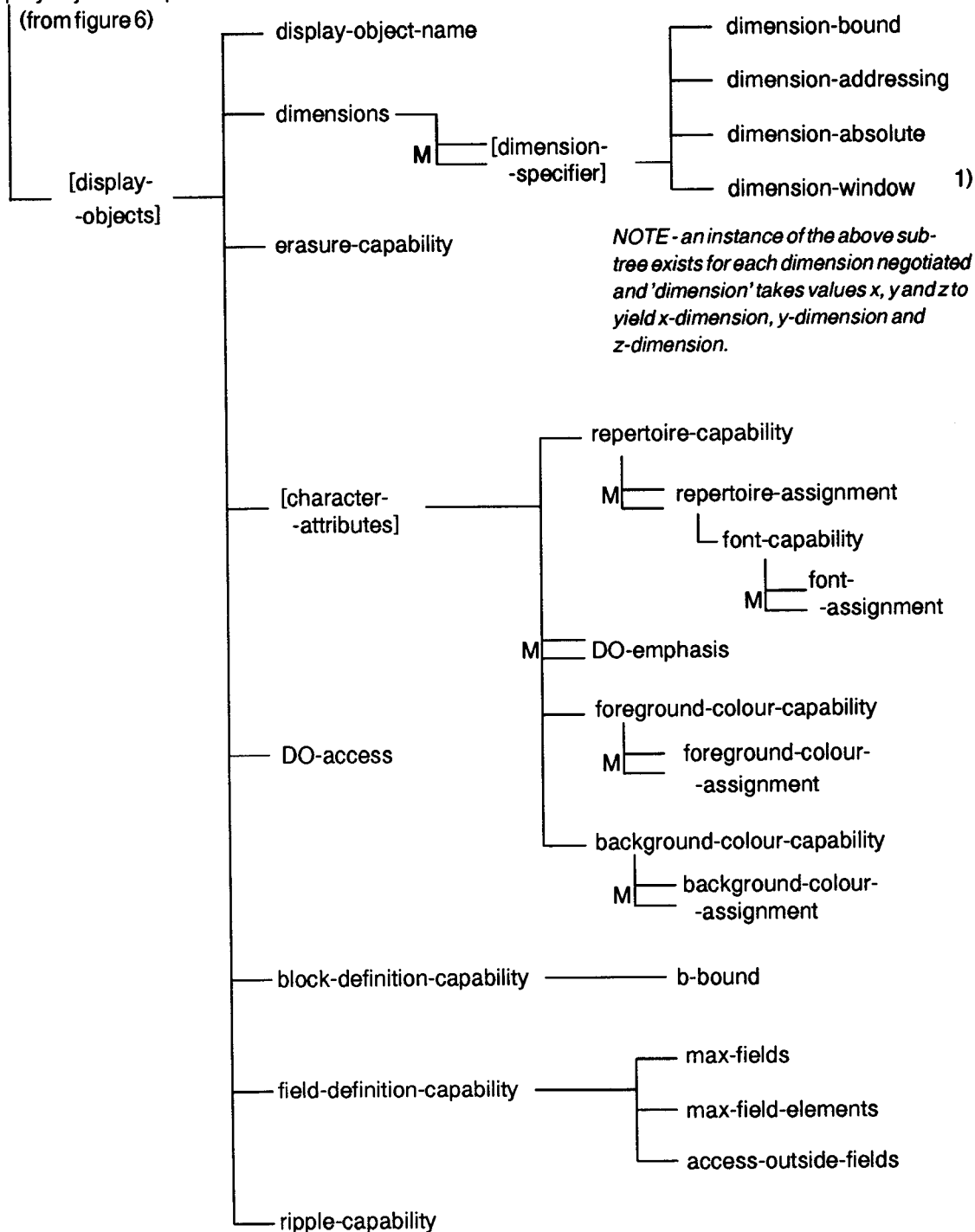


Figure 6 - Directed graph of VTE-parameters - part 1

Expansion of sub-tree for
display object VTE-parameters
(from figure 6)



1) When either block-definition-capability or field-definition-capability has value "yes" for a DO the VTE-parameter dimension-window has an explicit value only for the Z dimension.

Figure 7 - Directed graph of VTE-parameters - part 2